Advanced Data Analysis from an Elementary Point of View

Cosma Rohilla Shalizi

For my parents
and in memory of my grandparents

Contents

Introduction 12

Introduction 12
To the Reader 12
Concepts You Should Know 15

Part I Regression and Its Generalizations 17

1 Regression Basics 19
1.1 Statistics, Data Analysis, Regression 19
1.2 Guessing the Value of a Random Variable 20
1.3 The Regression Function 21
1.4 Estimating the Regression Function 25
1.5 Linear Smoothers 33
1.6 Further Reading 43
Exercises 43

2 The Truth about Linear Regression 45
2.1 Optimal Linear Prediction: Multiple Variables 45
2.2 Shifting Distributions, Omitted Variables, and Transformations 52
2.3 Adding Probabilistic Assumptions 61
2.4 Linear Regression Is Not the Philosopher's Stone 64
2.5 Further Reading 66
Exercises 67

3 Model Evaluation 68
3.1 What Are Statistical Models For? 68
3.2 Errors, In and Out of Sample 69
3.3 Over-Fitting and Model Selection 74
3.4 Cross-Validation 81
3.5 Warnings 89
3.6 Further Reading 93
Exercises 93

4 Smoothing in Regression 95
4.1 How Much Should We Smooth? 95

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

4.2 Adapting to Unknown Roughness 98
4.3 Kernel Regression with Multiple Inputs 110
4.4 Interpreting Smoothers: Plots 114
4.5 Average Predictive Comparisons 116
4.6 Computational Advice: `npreg` 118
4.7 Further Reading 124
Exercises 124

5 Simulation 125
5.1 What Is a Simulation? 125
5.2 How Do We Simulate Stochastic Models? 126
5.3 Repeating Simulations 130
5.4 Why Simulate? 131
5.5 Further Reading 137
Exercises 138

6 The Bootstrap 139
6.1 Stochastic Models, Uncertainty, Sampling Distributions 139
6.2 The Bootstrap Principle 141
6.3 Resampling 155
6.4 Bootstrapping Regression Models 158
6.5 Bootstrap with Dependent Data 164
6.6 Confidence Bands for Nonparametric Regression 164
6.7 Things Bootstrapping Does Poorly 164
6.8 Which Bootstrap When? 166
6.9 Further Reading 167
Exercises 168

7 Splines 169
7.1 Smoothing by Penalizing Curve Flexibility 169
7.2 Computational Example: Splines for Stock Returns 171
7.3 Basis Functions and Degrees of Freedom 177
7.4 Splines in Multiple Dimensions 179
7.5 Smoothing Splines versus Kernel Regression 180
7.6 Some of the Math Behind Splines 180
7.7 Further Reading 182
Exercises 183

8 Additive Models 185
8.1 Additive Models 185
8.2 Partial Residuals and Back-fitting 186
8.3 The Curse of Dimensionality 189
8.4 Example: California House Prices Revisited 192
8.5 Interaction Terms and Expansions 204
8.6 Closing Modeling Advice 206
8.7 Further Reading 206
Exercises 207

9 Testing Regression Specifications 209
9.1 Testing Functional Forms 209
9.2 Why Use Parametric Models At All? 220
9.3 Further Reading 224

10 Weighting and Variance 225
10.1 Weighted Least Squares 225
10.2 Heteroskedasticity 228
10.3 Estimating Conditional Variance Functions 239
10.4 Re-sampling Residuals with Heteroskedasticity 248
10.5 Local Linear Regression 249
10.6 Further Reading 256
Exercises 256

11 Logistic Regression 257
11.1 Modeling Conditional Probabilities 257
11.2 Logistic Regression 258
11.3 Numerical Optimization of the Likelihood 264
11.4 Generalized Linear and Generalized Additive Models 266
11.5 Model Checking 267
11.6 A Toy Example 269
11.7 Weather Forecasting in Snoqualmie Falls 274
11.8 Logistic Regression with More Than Two Classes 287
Exercises 288

12 GLMs and GAMs 290
12.1 Generalized Linear Models and Iterative Least Squares 290
12.2 Generalized Additive Models 296
12.3 Further Reading 296
Exercises 296

13 Trees 297
13.1 Prediction Trees 297
13.2 Regression Trees 300
13.3 Classification Trees 310
13.4 Further Reading 315
Exercises 315

Part II Distributions and Latent Structure 321

14 Density Estimation 323
14.1 Histograms Revisited 323
14.2 "The Fundamental Theorem of Statistics" 324
14.3 Error for Density Estimates 325
14.4 Kernel Density Estimates 328
14.5 Conditional Density Estimation 334
14.6 More on the Expected Log-Likelihood Ratio 335

14.7 Simulating from Density Estimates 338
14.8 Further Reading 343
Exercises 345

15 Principal Components Analysis 347
15.1 Mathematics of Principal Components 347
15.2 Example 1: Cars 354
15.3 Example 2: The United States *circa* 1977 358
15.4 Latent Semantic Analysis 361
15.5 PCA for Visualization 364
15.6 PCA Cautions 366
15.7 Random Projections 367
15.8 Further Reading 368
Exercises 369

16 Factor Models 372
16.1 From PCA to Factor Models 372
16.2 The Graphical Model 374
16.3 Roots of Factor Analysis in Causal Discovery 377
16.4 Estimation 379
16.5 The Rotation Problem 385
16.6 Factor Analysis as a Predictive Model 386
16.7 Factor Models versus PCA Once More 389
16.8 Examples in R 390
16.9 Reification, and Alternatives to Factor Models 394
16.10 Further Reading 401
Exercises 401

17 Mixture Models 403
17.1 Two Routes to Mixture Models 403
17.2 Estimating Parametric Mixture Models 407
17.3 Non-parametric Mixture Modeling 412
17.4 Worked Computating Example 412
17.5 Further Reading 422
Exercises 422

18 Graphical Models 432
18.1 Conditional Independence and Factor Models 432
18.2 Directed Acyclic Graph (DAG) Models 433
18.3 Conditional Independence and D-Separation 435
18.4 Independence and Information 442
18.5 Examples of DAG Models and Their Uses 444
18.6 Non-DAG Graphical Models 446
18.7 Further Reading 450
Exercises 451

Part III Causal Inference 453

19 Graphical Causal Models 455
19.1 Causation and Counterfactuals 455
19.2 Causal Graphical Models 456
19.3 Conditional Independence and d-Separation Revisited 459
19.4 Further Reading 460
Exercises 462

20 Identifying Causal Effects 463
20.1 Causal Effects, Interventions and Experiments 463
20.2 Identification and Confounding 465
20.3 Identification Strategies 467
20.4 Summary 482
Exercises 483

21 Estimating Causal Effects 485
21.1 Estimators in the Back- and Front- Door Criteria 485
21.2 Instrumental-Variables Estimates 492
21.3 Uncertainty and Inference 494
21.4 Recommendations 494
21.5 Further Reading 495
Exercises 496

22 Discovering Causal Structure 497
22.1 Testing DAGs 498
22.2 Testing Conditional Independence 499
22.3 Faithfulness and Equivalence 500
22.4 Causal Discovery with Known Variables 501
22.5 Software and Examples 506
22.6 Limitations on Consistency of Causal Discovery 512
22.7 Pseudo-code for the SGS Algorithm 512
22.8 Further Reading 513
Exercises 514

Part IV Dependent Data 515

23 Time Series 517
23.1 What Time Series Are 517
23.2 Stationarity 518
23.3 Markov Models 523
23.4 Autoregressive Models 527
23.5 Bootstrapping Time Series 532
23.6 Cross-Validation 534
23.7 Trends and De-Trending 534
23.8 Breaks in Time Series 539
23.9 Time Series with Latent Variables 540
23.10 Longitudinal Data 548
23.11 Multivariate Time Series 548

23.12 Further Reading 548
Exercises 550

24 Simulation-Based Inference 574
24.1 The Method of Simulated Moments 574
24.2 Indirect Inference 581
24.3 Further Reading 581
Exercises 582

Bibliography 582
References 583
Acknowledgments 605

Index 606
Index 607

Online Appendices 625

Appendix A **Big O and Little o Notation** 625

Appendix B **Taylor Expansions** 627

Appendix C **Propagation of Error** 630

Appendix D **Optimization** 632

Appendix E **Information Theory** 662

Appendix F **Relative Distributions and Smooth Tests** 681

Appendix G **Nonlinear Dimensionality Reduction** 707

Appendix H **Rudimentary Graph Theory** 731

Appendix I **Missing Data** 734

Appendix J **Programming** 762

Data-Analysis Assignments 3

1 Your Daddy's Rich 4

2 . . . But We Make It Up in Volume 9

3	**Past Performance, Future Results**	12
4	**Free Soil**	15
5	**There Were Giants in the Earth in Those Day**	18
6	**The Sound of Gunfire, Off in the Distance**	22
7	**The Bullet or the Ballot?**	25
8	**A Diversified Portfolio**	30
9	**The Monkey's Paw**	33
10	**What's That Got to Do with the Price of Condos in California?**	38
11	**The Advantages of Backwardness**	41
12	**It's Not the Heat that Gets You**	45
13	**Nice Demo City, but Will It Scale?**	48
14	**Fair's Affairs**	55
15	**How the North American Paleofauna Got a Crook in Its Regression Line**	57
16	**How the *Hyracotherium* Got Its Mass**	61
17	**How the Recent Mammals Got Their Size Distribution**	64
18	**Red Brain, Blue Brain**	67
19	**Brought to You by the Letters D, A and G**	70
20	**Teacher, Leave Those Kids Alone! (They're the Control Group)**	75
21	**Estimating with DAGs**	78
22	**Use and Abuse of Conditioning**	82
23	**What Makes the Union Strong?**	84
24	**An Insufficiently Random Walk Down Wall Street**	88
25	**Predicting Nine of the Last Five Recessions**	92
26	**Debt Needs Time for What It Kills to Grow In**	94

27 How Tetracycline Came to Peoria 96

Introduction

To the Reader

This book began as the notes for 36-402, Advanced Data Analysis, at Carnegie Mellon University. This is the methodological capstone of the core statistics sequence taken by our undergraduate majors (usually in their third year), and by undergraduate and graduate students from a range of other departments. The pre-requisite for that course is our class in modern linear regression, which in turn requires students to have taken classes in introductory statistics and data analysis, probability theory, mathematical statistics, linear algebra, and multivariable calculus. This book does not presume that you once learned but have forgotten that material; it presumes that you *know* those subjects and are ready to go further (see p. 15, at the end of this introduction). The book also presumes that you can read and write simple functions in R. If you are lacking in any of these areas, this book is not really for you, at least not now.

ADA is a class in *statistical methodology*: its aim is to get students to understand something of the range of modern[1] methods of data analysis, and of the considerations which go into choosing the right method for the job at hand (rather than distorting the problem to fit the methods you happen to know). Statistical *theory* is kept to a minimum, and largely introduced as needed. Since ADA is also a class in *data analysis*, there are a lot of assignments in which large, real data sets are analyzed with the new methods.

There is no way to cover every important topic for data analysis in just a semester. Much of what's not here — sampling theory and survey methods, experimental design, advanced multivariate methods, hierarchical models, the intricacies of categorical data, graphics, data mining, spatial and spatio-temporal statistics — gets covered by our other undergraduate classes. Other important areas, like networks, inverse problems, advanced model selection or robust estimation, have to wait for graduate school[2].

The mathematical level of these notes is deliberately low; nothing should be beyond a competent third-year undergraduate. But every subject covered here can be profitably studied using vastly more sophisticated techniques; that's why

[1] Just as an undergraduate "modern physics" course aims to bring the student up to about 1930 (more specifically, to 1926), this class aims to bring the student up to about 1990–1995, maybe 2000.

[2] Early drafts of this book, circulated online, included sketches of chapters covering spatial statistics, networks, and experiments. These were all sacrificed to length, and to actually finishing.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

this is advanced data analysis from an *elementary* point of view. If reading these pages inspires anyone to study the same material from an *advanced* point of view, I will consider my troubles to have been amply repaid.

A final word. At this stage in your statistical education, you have gained two kinds of knowledge — a few general statistical principles, and many more specific procedures, tests, recipes, etc. Typical students are much more comfortable with the specifics than the generalities. But the truth is that while none of your recipes are *wrong*, they are tied to assumptions which hardly ever hold[3]. Learning more flexible and powerful methods, which have a much better hope of being reliable, will demand a lot of hard thinking and hard work. Those of you who succeed, however, will have done something you can be proud of.

Organization of the Book

Part I is about regression and its generalizations. The focus is on nonparametric regression, especially smoothing methods. (Chapter 2 motivates this by dispelling some myths and misconceptions about linear regression.) The ideas of cross-validation, of simulation, and of the bootstrap all arise naturally in trying to come to grips with regression. This part also covers classification and specification-testing.

Part II is about learning distributions, especially multivariate distributions, rather than doing regression. It is possible to learn essentially arbitrary distributions from data, including conditional distributions, but the number of observations needed is often prohibitive when the data is high-dimensional. This motivates looking for models of special, simple structure lurking behind the high-dimensional chaos, including various forms of linear and non-linear dimension reduction, and mixture or cluster models. All this builds towards the general idea of using graphical models to represent dependencies between variables.

Part III is about causal inference. This is done entirely within the graphical-model formalism, which makes it easy to understand the difference between causal prediction and the more ordinary "actuarial" prediction we are used to as statisticians. It also greatly simplifies figuring out when causal effects are, or are not, identifiable from our data. (Among other things, this gives us a sound way to decide what we ought to control for.) Actual estimation of causal effects is done as far as possible non-parametrically. This part ends by considering procedures for discovering causal structure from observational data.

Part IV moves away from independent observations, more or less tacitly as-

[3] "Econometric theory is like an exquisitely balanced French recipe, spelling out precisely with how many turns to mix the sauce, how many carats of spice to add, and for how many milliseconds to bake the mixture at exactly 474 degrees of temperature. But when the statistical cook turns to raw materials, he finds that hearts of cactus fruit are unavailable, so he substitutes chunks of cantaloupe; where the recipe calls for vermicelli he uses shredded wheat; and he substitutes green garment dye for curry, ping-pong balls for turtle's eggs and, for Chalifougnac vintage 1883, a can of turpentine." — Stefan Valavanis, quoted in Roger Koenker, "Dictionary of Received Ideas of Statistics" (`http://www.econ.uiuc.edu/~roger/dict.html`), s.v. "Econometrics".

sumed earlier, to dependent data. It specifically considers models of time series, and time series data analysis, and simulation-based inference for complex or analytically-intractable models.

Parts III and IV are mostly independent of each other, but both rely on Parts I and II.

The online appendices contain a number of optional topics omitted from the main text in the interest of length, some mathematical reminders, and advice on writing R code for data analysis.

R Examples

The book is full of worked computational examples in R. In most cases, the code used to make figures, tables, etc., is given in full in the text. (The code is deliberately omitted for a few examples for pedagogical reasons.) To save space, comments are generally omitted from the text, but comments are vital to good programming (§J.9.1), so fully-commented versions of the code for each chapter are available from the book's website.

Problems

There are two kinds of problems included here. Mathematical and computational exercises go at the end of chapters, since they are mostly connected to those pieces of content. (Many of them are complements to, or fill in details of, material in the chapters.) There are also data-centric assignments, consisting of extended problem sets, in the companion document. Most of these draw on material from multiple chapters, and many of them are based on specific papers.

Solutions will be available to teachers from the publisher; giving them out to those using the book for self-study is, sadly, not feasible.

To Teachers

The usual one-semester course for this class has contained Chapters 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22 and 23, and Appendix and J (the latter quite early on). Other chapters and appendices have rotated in and out from year to year. One of the problem sets from Appendix 24.3 (or a similar one) was due every week, either as homework or as a take-home exam.

Corrections and Updates

The page for this book is `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`. The latest version will live there. The book will eventually be published by Cambridge University Press, at which point there will still be a free next-to-final draft at that URL, and errata. While the book is still in a draft, the PDF contains notes to myself for revisions, [[like so]]; you can ignore them.

[[Also marginal notes-to-self]]

Concepts You Should Know

If more than a few of these are unfamiliar, it's unlikely you're ready for this book.

Linear algebra: Vectors; arithmetic with vectors; inner or dot product of vectors, orthogonality; linear independence; basis vectors. Linear subspaces. Matrices, matrix arithmetic, multiplying vectors and matrices; geometric meaning of matrix multiplication. Eigenvalues and eigenvectors of matrices. Projection.

Calculus: Derivative, integral; fundamental theorem of calculus. Multivariable extensions: gradient, Hessian matrix, multidimensional integrals. Finding minima and maxima with derivatives. Taylor approximations (App. B).

Probability: Random variable; distribution, population, sample. Cumulative distribution function, probability mass function, probability density function. Specific distributions: Bernoulli, binomial, Poisson, geometric, Gaussian, exponential, t, Gamma. Expectation value. Variance, standard deviation.

Joint distribution functions. Conditional distributions; conditional expectations and variances. Statistical independence and dependence. Covariance and correlation; why dependence is not the same thing as correlation. Rules for arithmetic with expectations, variances and covariances. Laws of total probability, total expectation, total variation. Sequences of random variables. Stochastic process. Law of large numbers. Central limit theorem.

Statistics: Sample mean, sample variance. Median, mode. Quartile, percentile, quantile. Inter-quartile range. Histograms. Contingency tables; odds ratio, log odds ratio.

Parameters; estimator functions and point estimates. Sampling distribution. Bias of an estimator. Standard error of an estimate; standard error of the mean; how and why the standard error of the mean differs from the standard deviation. Consistency of estimators. Confidence intervals and interval estimates.

Hypothesis tests. Tests for differences in means and in proportions; Z and t tests; degrees of freedom. Size, significance, power. Relation between hypothesis tests and confidence intervals. χ^2 test of independence for contingency tables; degrees of freedom. KS test for goodness-of-fit to distributions.

Likelihood. Likelihood functions. Maximum likelihood estimates. Relation between confidence intervals and the likelihood function. Likelihood ratio test.

Regression: What a linear model is; distinction between the regressors and the regressand. Predictions/fitted values and residuals of a regression. Interpretation of regression coefficients. Least-squares estimate of coefficients. Relation between maximum likelihood, least squares, and Gaussian distributions. Matrix formula for estimating the coefficients; the hat matrix for finding fitted values. R^2; why adding more predictor variables never reduces R^2. The t-test for the significance of individual coefficients given other coefficients. The F-test and partial F-test for the significance of groups of coefficients. Degrees of freedom for residuals. Diagnostic examination of residuals. Confidence intervals for parameters. Confidence intervals for fitted values. Prediction intervals. (Most of this material is reviewed at `http://www.stat.cmu.edu/~cshalizi/TALR/`.)

Part I

Regression and Its Generalizations

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

1

Regression: Predicting and Relating Quantitative Features

1.1 Statistics, Data Analysis, Regression

Statistics is the branch of mathematical engineering which designs and analyses methods for drawing reliable inferences from imperfect data.

The subject of most sciences is some aspect of the world around us, or within us. Psychology studies minds; geology studies the Earth's composition and form; economics studies production, distribution and exchange; mycology studies mushrooms. Statistics does not study the world, but some of the ways we try to understand the world — some of the intellectual tools of the other sciences. Its utility comes indirectly, through helping those other sciences.

This utility is very great, because all the sciences have to deal with imperfect data. Data may be imperfect because we can only observe and record a small fraction of what is relevant; or because we can only observe indirect signs of what is truly relevant; or because, no matter how carefully we try, our data always contain an element of noise. Over the last two centuries, statistics has come to handle all such imperfections by modeling them as random processes, and probability has become so central to statistics that we introduce random events deliberately (as in sample surveys).[1]

Statistics, then, uses probability to model inference from data. We try to mathematically understand the properties of different procedures for drawing inferences: Under what conditions are they reliable? What sorts of errors do they make, and how often? What can they tell us when they work? What are signs that something has gone wrong? Like other branches of engineering, statistics aims not just at understanding but also at improvement: we want to analyze data *better*: more reliably, with fewer and smaller errors, under broader conditions, faster, and with less mental effort. Sometimes some of these goals conflict — a fast, simple method might be very error-prone, or only reliable under a narrow range of circumstances.

One of the things that people most often want to know about the world is how different variables are related to each other, and one of the central tools statistics has for learning about relationships is regression.[2] In your linear regression class,

[1] Two excellent, but very different, histories of how statistics came to this understanding are Hacking (1990) and Porter (1986).

[2] The origin of the name is instructive (Stigler, 1986). It comes from 19th century investigations into the relationship between the attributes of parents and their children. People who are taller (heavier, faster, ...) than average tend to have children who are also taller than average, but not *quite* as tall.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

you learned about how it could be used in data analysis, and learned about its properties. In this book, we will build on that foundation, extending beyond basic linear regression in many directions, to answer many questions about how variables are related to each other.

This is intimately related to prediction. Being able to make predictions isn't the *only* reason we want to understand relations between variables — we also want to answer "what if?" questions — but prediction *tests* our knowledge of relations. (If we misunderstand, we might still be able to predict, but it's hard to see how we could understand and *not* be able to predict.) So before we go beyond linear regression, we will first look at prediction, and how to predict one variable from nothing at all. Then we will look at predictive relationships between variables, and see how linear regression is just one member of a big family of smoothing methods, all of which are available to us.

1.2 Guessing the Value of a Random Variable

We have a quantitative, numerical variable, which we'll imaginatively call Y. We'll suppose that it's a random variable, and try to predict it by guessing a single value for it. (Other kinds of predictions are possible — we might guess whether Y will fall within certain limits, or the probability that it does so, or even the whole probability distribution of Y. But some lessons we'll learn here will apply to these other kinds of predictions as well.) What is the best value to guess? More formally, what is the **optimal point forecast** for Y?

[[TODO: Enforce a strict distinction in symbols and names between in-sample MSE and expected squared error (maybe "ESE"?). Or is that too confusing to students?]]

To answer this question, we need to pick a function to be optimized, which should measure how good our guesses are — or equivalently how bad they are, i.e., how big an error we're making. A reasonable, traditional starting point is the **mean squared error**:

$$\mathrm{MSE}(m) \equiv \mathbb{E}\left[(Y-m)^2\right] \tag{1.1}$$

So we'd like to find the value μ where $\mathrm{MSE}(m)$ is smallest. Start by re-writing the MSE as a (squared) bias plus a variance:

$$\begin{aligned}
\mathrm{MSE}(m) &= \mathbb{E}\left[(Y-m)^2\right] && (1.2)\\
&= \left(\mathbb{E}\left[Y-m\right]\right)^2 + \mathbb{V}\left[Y-m\right] && (1.3)\\
&= \left(\mathbb{E}\left[Y-m\right]\right)^2 + \mathbb{V}\left[Y\right] && (1.4)\\
&= \left(\mathbb{E}\left[Y\right]-m\right)^2 + \mathbb{V}\left[Y\right] && (1.5)
\end{aligned}$$

Notice that only the first, bias-squared term depends on our prediction m. We want to find the derivative of the MSE with respect to our prediction m, and

Likewise, the children of unusually short parents also tend to be closer to the average, and similarly for other traits. This came to be called "regression towards the mean," or even "regression towards mediocrity"; hence the line relating the average height (or whatever) of children to that of their parents was "the regression line," and the word stuck.

then set that to zero at the optimal prediction μ:

$$\frac{d\text{MSE}}{dm} = -2\left(\mathbb{E}\left[Y\right] - m\right) + 0 \tag{1.6}$$

$$\left.\frac{d\text{MSE}}{dm}\right|_{m=\mu} = 0 \tag{1.7}$$

$$2(\mathbb{E}\left[Y\right] - \mu) = 0 \tag{1.8}$$

$$\mu = \mathbb{E}\left[Y\right] \tag{1.9}$$

So, if we gauge the quality of our prediction by mean-squared error, the best prediction to make is the expected value.

1.2.1 Estimating the Expected Value

Of course, to make the prediction $\mathbb{E}\left[Y\right]$ we would have to know the expected value of Y. Typically, we do not. However, if we have sampled values, $y_1, y_2, \ldots y_n$, we can estimate the expectation from the sample mean:

$$\widehat{\mu} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i \tag{1.10}$$

If the samples are independent and identically distributed (IID), then the law of large numbers tells us that

$$\widehat{\mu} \to \mathbb{E}\left[Y\right] = \mu \tag{1.11}$$

and algebra with variances (Exercise 1.1) tells us something about how fast the convergence is, namely that the squared error will typically be $\mathbb{V}\left[Y\right]/n$.

Of course the assumption that the y_i come from IID samples is a strong one, but we can assert pretty much the same thing if they're just uncorrelated with a common expected value. Even if they are correlated, but the correlations decay fast enough, all that changes is the rate of convergence (§23.2.2.1). So "sit, wait, and average" is a pretty reliable way of estimating the expectation value.

1.3 The Regression Function

Of course, it's not very useful to predict *just one number* for a variable. Typically, we have lots of variables in our data, and we believe they are related *somehow*. For example, suppose that we have data on two variables, X and Y, which might look like Figure 1.1.[3] The feature Y is what we are trying to predict, a.k.a. the **dependent variable** or **output** or **response** or **regressand**, and X is the **predictor** or **independent variable** or **covariate** or **input** or **regressor**. Y might be something like the profitability of a customer and X their credit rating, or, if you want a less mercenary example, Y could be some measure of improvement in blood cholesterol and X the dose taken of a drug. Typically we

[3] Problem set 27 features data that looks rather like these made-up values.

won't have just one input feature X but rather many of them, but that gets harder to draw and doesn't change the points of principle.

Figure 1.2 shows the same data as Figure 1.1, only with the sample mean added on. This clearly tells us something about the data, but also it seems like we should be able to do better — to reduce the average error — by using X, rather than by ignoring it.

Let's say that the we want our prediction to be a *function* of X, namely $f(X)$. What should that function be, if we still use mean squared error? We can work this out by using the law of total expectation, i.e., the fact that $\mathbb{E}[U] = \mathbb{E}[\mathbb{E}[U|V]]$ for any random variables U and V.

$$\begin{aligned}
\mathrm{MSE}(f) &= \mathbb{E}\left[(Y - f(X))^2\right] && (1.12)\\
&= \mathbb{E}\left[\mathbb{E}\left[(Y - f(X))^2|X\right]\right] && (1.13)\\
&= \mathbb{E}\left[\mathbb{V}[Y - f(X)|X] + (\mathbb{E}[Y - f(X)|X])^2\right] && (1.14)\\
&= \mathbb{E}\left[\mathbb{V}[Y|X] + (\mathbb{E}[Y - f(X)|X])^2\right] && (1.15)
\end{aligned}$$

When we want to minimize this, the first term inside the expectation doesn't depend on our prediction, and the second term looks just like our previous optimization only with all expectations conditional on X. So our optimal function $\mu(x)$ is

$$\mu(x) = \mathbb{E}[Y|X = x] \tag{1.16}$$

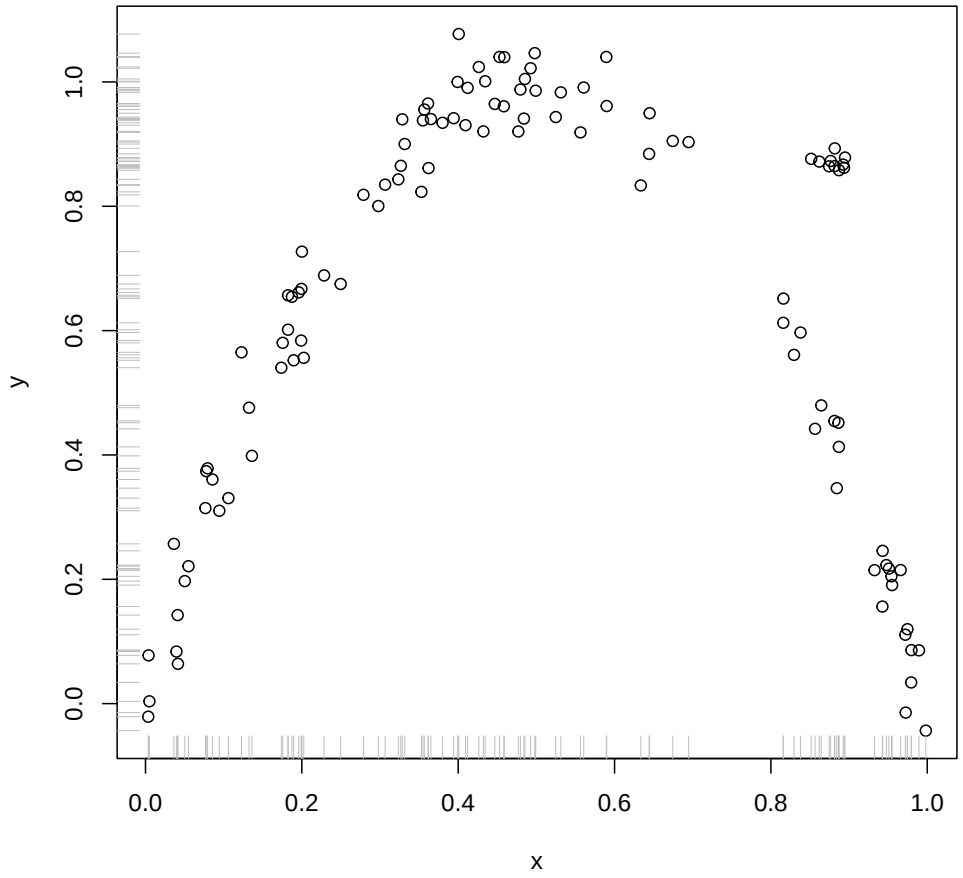
In other words, the (mean-squared) optimal *conditional* prediction is just the conditional expected value. The function $\mu(x)$ is called the **true regression function**, the **optimal regression function**, the **population regression function**, or just the **regression function**. This is what we would like to know when we want to predict Y.

Some Disclaimers

It's important to be clear on what is and is not being assumed here. Talking about X as the "independent variable" and Y as the "dependent" one suggests a causal model, which we might write

$$Y \leftarrow \mu(X) + \epsilon \tag{1.17}$$

where the direction of the arrow, $\leftarrow$, indicates the flow from causes to effects, and ϵ is some noise variable. If the gods of inference are very kind, then ϵ would have a fixed distribution, independent of X, and we could without loss of generality take it to have mean zero. ("Without loss of generality" because if it has a non-zero mean, we can incorporate that into $\mu(X)$ as an additive constant.) However, *no* such assumption is required to get Eq. 1.16. It works when predicting effects from causes, or the other way around when predicting (or "retrodicting") causes from effects, or indeed when there is no causal relationship whatsoever between X and



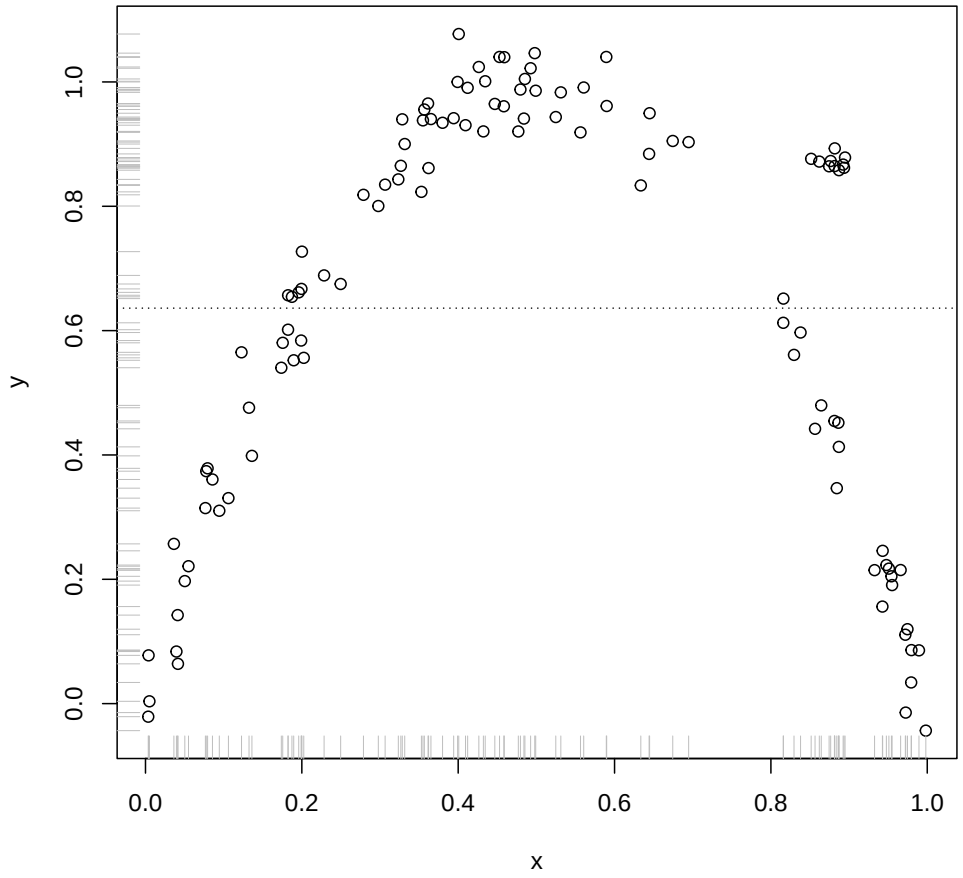

```
plot(all.x, all.y, xlab = "x", ylab = "y")
rug(all.x, side = 1, col = "grey")
rug(all.y, side = 2, col = "grey")
```

Figure 1.1 Scatterplot of the (made up) running example data. `rug()` adds horizontal and vertical ticks to the axes to mark the location of the data; this isn't necessary but is often helpful. The data are in the `basics-examples.Rda` file.

Y[4]. It *is* always true that

$$Y|X = \mu(X) + \epsilon(X) \tag{1.18}$$

[4] We will cover causal inference in detail in Part III.




```
plot(all.x, all.y, xlab = "x", ylab = "y")
rug(all.x, side = 1, col = "grey")
rug(all.y, side = 2, col = "grey")
abline(h = mean(all.y), lty = "dotted")
```

Figure 1.2 Data from Figure 1.1, with a horizontal line at $\overline{y}$.

where $\epsilon(X)$ is a random variable with expected value 0, $\mathbb{E}[\epsilon|X = x] = 0$, but as the notation indicates the distribution of this variable generally depends on X.

It's also important to be clear that if we find the regression function is a constant, $\mu(x) = \mu_0$ for all x, that this does not mean that X and Y are statistically

independent. If they are independent, then the regression function is a constant, but turning this around is the logical fallacy of "affirming the consequent"[5].

1.4 Estimating the Regression Function

We want the regression function $\mu(x) = \mathbb{E}[Y|X = x]$, but what we have is a pile of training examples, of pairs $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$. What should we do?

If X takes on only a finite set of values, then a simple strategy is to use the conditional sample means:

$$\widehat{\mu}(x) = \frac{1}{\#\{i : x_i = x\}} \sum_{i:x_i=x} y_i \tag{1.19}$$

Reasoning with the law of large numbers as before, we can be confident that $\widehat{\mu}(x) \to \mathbb{E}[Y|X = x]$.

Unfortunately, this *only* works when X takes values in a finite set. If X is continuous, then in general the probability of our getting a sample at any *particular* value is zero, as is the probability of getting *multiple* samples at *exactly* the same value of x. This is a basic issue with estimating any kind of function from data — the function will always be **undersampled**, and we need to fill in between the values we see. We also need to somehow take into account the fact that each y_i is a *sample* from the conditional distribution of $Y|X = x_i$, and generally not equal to $\mathbb{E}[Y|X = x_i]$. So any kind of function estimation is going to involve interpolation, extrapolation, and de-noising or smoothing.

Different methods of estimating the regression function — different regression methods, for short — involve different choices about how we interpolate, extrapolate and smooth. These are choices about how to approximate $\mu(x)$ with a limited class of functions which we know (or at least hope) we can estimate. There is no guarantee that our choice leads to a *good* approximation in the case at hand, though it is sometimes possible to say that the approximation error will shrink as we get more and more data. This is an extremely important topic and deserves an extended discussion, coming next.

1.4.1 The Bias-Variance Trade-off

Suppose that the true regression function is $\mu(x)$, but we use the function $\widehat{\mu}$ to make our predictions. Let's look at the mean squared error at $X = x$ in a slightly different way than before, which will make it clearer what happens when we can't use μ to make predictions. We'll begin by expanding $(Y - \widehat{\mu}(x))^2$, since the MSE at x is just the expectation of this.

$$(Y - \widehat{\mu}(x))^2 \tag{1.20}$$

$$= (Y - \mu(x) + \mu(x) - \widehat{\mu}(x))^2$$

$$= (Y - \mu(x))^2 + 2(Y - \mu(x))(\mu(x) - \widehat{\mu}(x)) + (\mu(x) - \widehat{\mu}(x))^2 \tag{1.21}$$

[5] As in combining the fact that all human beings are featherless bipeds, and the observation that a cooked turkey is a featherless biped, to conclude that cooked turkeys are human beings.

Eq. 1.18 tells us that $Y - \mu(X) = \epsilon$, a random variable which has expectation zero (and is uncorrelated with X). Taking the expectation of Eq. 1.21, nothing happens to the last term (since it doesn't involve any random quantities); the middle term goes to zero (because $\mathbb{E}[Y - \mu(X)] = \mathbb{E}[\epsilon] = 0$), and the first term becomes the variance of ϵ, call it $\sigma^2(x)$:

$$\mathrm{MSE}(\widehat{\mu}(x)) = \sigma^2(x) + (\mu(x) - \widehat{\mu}(x))^2 \tag{1.22}$$

The $\sigma^2(x)$ term doesn't depend on our prediction function, just on how hard it is, intrinsically, to predict Y at $X = x$. The second term, though, is the extra error we get from not knowing μ. (Unsurprisingly, ignorance of μ cannot *improve* our predictions.) This is our first **bias-variance decomposition**: the total MSE at x is decomposed into a (squared) bias $\mu(x) - \widehat{\mu}(x)$, the amount by which our predictions are *systematically* off, and a variance $\sigma^2(x)$, the unpredictable, "statistical" fluctuation around even the best prediction.

All this presumes that $\widehat{\mu}$ is a single fixed function. Really, of course, $\widehat{\mu}$ is something we estimate from earlier data. But if those data are random, the regression function we get is random too; let's call this random function $\widehat{M}_n$, where the subscript reminds us of the finite amount of data we used to estimate it. What we have analyzed is really $\mathrm{MSE}(\widehat{M}_n(x)|\widehat{M}_n = \widehat{\mu})$, the mean squared error *conditional on* a particular estimated regression function. What can we say about the prediction error of the *method*, averaging over all the possible training data sets?

$$\begin{aligned}
\mathrm{MSE}(\widehat{M}_n(x)) &= \mathbb{E}\left[(Y - \widehat{M}_n(X))^2 | X = x\right] && (1.23)\\
&= \mathbb{E}\left[\mathbb{E}\left[(Y - \widehat{M}_n(X))^2 | X = x, \widehat{M}_n = \widehat{\mu}\right] | X = x\right] && (1.24)\\
&= \mathbb{E}\left[\sigma^2(x) + (\mu(x) - \widehat{M}_n(x))^2 | X = x\right] && (1.25)\\
&= \sigma^2(x) + \mathbb{E}\left[(\mu(x) - \widehat{M}_n(x))^2 | X = x\right] && (1.26)\\
&= \sigma^2(x) + \mathbb{E}\left[(\mu(x) - \mathbb{E}\left[\widehat{M}_n(x)\right] + \mathbb{E}\left[\widehat{M}_n(x)\right] - \widehat{M}_n(x))^2\right] && (1.27)\\
&= \sigma^2(x) + \left(\mu(x) - \mathbb{E}\left[\widehat{M}_n(x)\right]\right)^2 + \mathbb{V}\left[\widehat{M}_n(x)\right] && (1.28)
\end{aligned}$$

This is our second bias-variance decomposition — I pulled the same trick as before, adding and subtracting a mean inside the square. The first term is just the variance of the process; we've seen that before and it isn't, for the moment, of any concern. The second term is the bias in using $\widehat{M}_n$ to estimate μ — the **approximation bias** or **approximation error**. The third term, though, is the variance in our *estimate* of the regression function. Even if we have an unbiased *method* ($\mu(x) = \mathbb{E}\left[\widehat{M}_n(x)\right]$), if there is a lot of variance in our estimates, we can expect to make large errors.

The approximation bias depends on the true regression function. For example, if $\mathbb{E}\left[\widehat{M}_n(x)\right] = 42 + 37x$, the error of approximation will be zero at all x if $\mu(x) = 42 + 37x$, but it will be larger and x-dependent if $\mu(x) = 0$. However, there are flexible methods of estimation which will have small approximation biases for

all μ in a broad range of regression functions. The catch is that, at least past a certain point, decreasing the approximation bias can only come through increasing the estimation variance. This is the **bias-variance trade-off**. However, nothing says that the trade-off has to be one-for-one. Sometimes we can lower the total error by *introducing* some bias, since it gets rid of more variance than it adds approximation error. The next section gives an example.

In general, both the approximation bias and the estimation variance depend on n. A method is **consistent**[6] when both of these go to zero as $n \to \infty$ — that is, if we recover the true regression function as we get more and more data.[7] Again, consistency depends not just on the method, but also on how well the method matches the data-generating process, and, again, there is a bias-variance trade-off. There can be multiple consistent methods for the same problem, and their biases and variances don't have to go to zero at the same *rates*.

1.4.2 The Bias-Variance Trade-Off in Action

Let's take an extreme example: we could decide to approximate $\mu(x)$ by a constant μ_0. The implicit smoothing here is very strong, but sometimes appropriate. For instance, it's appropriate when $\mu(x)$ really is a constant! Then trying to estimate any additional structure in the regression function is just wasted effort. Alternately, if $\mu(x)$ is *nearly* constant, we may still be better off approximating it as one. For instance, suppose the true $\mu(x) = \mu_0 + a\sin(\nu x)$, where $a \ll 1$ and $\nu \gg 1$ (Figure 1.3 shows an example). With limited data, we can actually get better predictions by estimating a constant regression function than one with the correct functional form.

1.4.3 Ordinary Least Squares Linear Regression as Smoothing

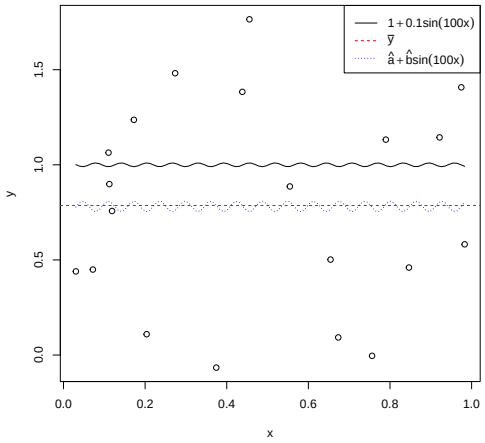
Let's revisit ordinary least-squares linear regression from this point of view. We'll assume that the predictor variable X is one-dimensional, just to simplify the book-keeping.

We *choose* to approximate $\mu(x)$ by $b_0 + b_1 x$, and ask for the best values β_0, β_1 of those constants. When I need to talk about the function $\beta_0 + \beta_1 x$, I'll write it as $\lambda(x)$, not $\mu(x)$, to emphasize that it's a linear approximation.

The coefficients in $\lambda(x)$ will be the ones which minimize the mean-squared

[6] To be precise, **consistent for** μ, or **consistent for conditional expectations**. More generally, an estimator of any property of the data, or of the whole distribution, is consistent if it converges on the truth.

[7] You might worry about this claim, especially if you've taken more probability theory — aren't we just saying something about average performance of the $\widehat{M}_n$, rather than any *particular* estimated regression function? But notice that if the estimation variance goes to zero, then by Chebyshev's inequality, $\Pr(|X - \mathbb{E}[X]| \geq a) \leq \mathbb{V}[X]/a^2$, each $\widehat{M}_n(x)$ comes arbitrarily close to $\mathbb{E}\left[\widehat{M}_n(x)\right]$ with arbitrarily high probability. If the approximation bias goes to zero, therefore, the estimated regression functions converge *in probability* on the true regression function, not just *in mean*.




```
ugly.func <- function(x) {
    1 + 0.01 * sin(100 * x)
}
x <- runif(20)
y <- ugly.func(x) + rnorm(length(x), 0, 0.5)
plot(x, y, xlab = "x", ylab = "y")
curve(ugly.func, add = TRUE)
abline(h = mean(y), col = "red", lty = "dashed")
sine.fit = lm(y ~ 1 + sin(100 * x))
curve(sine.fit$coefficients[1] + sine.fit$coefficients[2] * sin(100 * x), col = "blue",
    add = TRUE, lty = "dotted")
legend("topright", legend = c(expression(1 + 0.1 * sin(100 * x)), expression(bar(y)),
    expression(hat(a) + hat(b) * sin(100 * x))), lty = c("solid", "dashed", "dotted"),
    col = c("black", "red", "blue"))
```

Figure 1.3 When we try to estimate a rapidly-varying but small-amplitude regression function (solid black line, $\mu = 1 + 0.01 \sin 100x + \epsilon$, with mean-zero Gaussian noise of standard deviation 0.5), we can do better to use a constant function (red dashed line at the sample mean) than to estimate a more complicated model of the *correct* functional form $\hat{a} + \hat{b} \sin 100x$ (dotted blue line). With just 20 observations, the mean predicts slightly better on new data (square-root MSE, RMSE, of 0.54) than does the estimate sine function (RMSE of 0.55). The bias of using the wrong functional form is less than the extra variance of estimation, so using the true model form hurts us.

error. Start with the basic identity that $\mathbb{E}[Z^2] = \mathbb{V}[Z] - (\mathbb{E}[Z])^2$:

$$\begin{aligned} \mathrm{MSE}(a,b) &= \mathbb{E}\left[(Y - b_0 - b_1 X)^2\right] && (1.29)\\ &= \mathbb{V}[Y - b_0 - b_1 X] + (\mathbb{E}[Y - b_0 - b_1 X])^2 && (1.30)\\ &= \mathbb{V}[Y - b_1 X] + (\mathbb{E}[Y] - b_0 - b - 1\mathbb{E}[X])^2 && (1.31) \end{aligned}$$

since additive constants don't change variances. The first, variance term in Eq.

?? doesn't involve b_0 at all, just b_1; b_0 only matters for the second, squared-difference-in-expectations term.

It's now actually straightforward to find the optimal intercept β_0. The only term in which it appears is a square, which is necessarily ≥ 0. But we can make the square equal to zero at the optimum by using

$$\beta_0 = \mathbb{E}[Y] - \beta_1 \mathbb{E}[X] \tag{1.32}$$

(You can also get this by taking the derivative of Eq. 1.31 with respect to b_0 and setting it to zero at the optimum.)

What about the slope? There we really do need to differentiate. Remember that $\mathbb{V}[U+V] = \mathbb{V}[U] + \mathbb{V}[V] + 2\mathrm{Cov}[U,V]$, so

$$\mathbb{V}[Y - b_1 X] = \mathbb{V}[Y] + b_1^2 \mathbb{V}[X] - 2b_1 \mathrm{Cov}[X,Y] \tag{1.3}$$

$$\frac{\partial \mathbb{V}[Y - b_1 X]}{\partial b_1} = 0 + 2b_1 \mathbb{V}[X] - 2\mathrm{Cov}[X,Y] \tag{1.3}$$

$$\frac{\partial \left[\mathbb{V}[Y - b_1 X] + (\mathbb{E}[Y] - b_0 - b - 1\mathbb{E}[X])^2\right]}{\partial b_1} = 2b_1 \mathbb{V}[X] - 2\mathrm{Cov}[X,Y] - 2(\mathbb{E}[Y] - b_0 - b_1 \mathbb{E}[X])\mathbb{E}[X] \tag{1.3}$$

$$\therefore \beta_1 = \frac{\mathrm{Cov}[X,Y]}{\mathbb{V}[X]} \tag{1.3}$$

using Eq. refeqn:intercept-of-optimal-linear-predictor-1d.

Since the optimal linear model is $\beta_0 + \beta_1 X$, if we put together our expressions for the two coefficients, we see that the optimal *linear*-in-X prediction is

$$\lambda(x) = \mathbb{E}[Y] + \frac{\mathrm{Cov}[X,Y]}{\mathbb{V}[X]}(x - \mathbb{E}[X]) \tag{1.37}$$

It's worth pausing here to realize that we didn't make any assumption about whether μ is linear. Eq. 1.37 describes the (unique) optimal linear prediction function λ, whether or not $\lambda = \mu$. We also didn't assume anything about X being an "independent variable" and Y being a "dependent variable" in any important sense; that anything has a Gaussian distribution; that the noise has a constant distribution or even a constant variance. If we *decide* to use a linear prediction function, and want the best one of those, we're committed to Eq. 1.37[8].

Now, if we try to estimate this from data, there are (at least) two approaches. One is to replace the true, population values of the covariance and the variance with their sample values, respectively

$$\frac{1}{n}\sum_i (y_i - \overline{y})(x_i - \overline{x}) \tag{1.38}$$

and

$$\frac{1}{n}\sum_i (x_i - \overline{x})^2 \equiv \widehat{\mathbb{V}}[X]\,. \tag{1.39}$$

[8] Why most treatments of linear regression *do* add all those assumptions is another story, covered in Chapter **??**.

The other is to minimize the **in-sample** or **empirical** mean squared error,

$$\frac{1}{n}\sum_i (y_i - b_0 - b_1 x_i)^2 \tag{1.40}$$

You may or may not find it surprising that both approaches lead to the same answer:

$$\widehat{\beta_1} = \frac{\frac{1}{n}\sum_i (y_i - \overline{y})(x_i - \overline{x})}{\widehat{\mathbb{V}}[X]} \tag{1.41}$$

$$\widehat{\beta_0} = \overline{y} - \widehat{\beta_1}\overline{x} \tag{1.42}$$

$$\tag{1.43}$$

Provided that $\mathbb{V}[X] > 0$, these will converge with IID samples, so we have a consistent estimator.

We are now in a position to see how the least-squares linear regression model is really a weighted averaging of the data. Let's write the estimated linear prediction function function explicitly in terms of the training data points.

$$\widehat{\lambda}(x) = \widehat{\beta_0} + \widehat{\beta_1} x \tag{1.44}$$

$$= \overline{y} + \widehat{\beta_1}(x - \overline{x}) \tag{1.45}$$

$$= \frac{1}{n}\sum_{i=1}^{n} y_i + \left(\frac{\frac{1}{n}\sum_i (y_i - \overline{y})(x_i - \overline{x})}{\frac{1}{n}\sum_i (x_i - \overline{x})^2}\right)(x - \overline{x}) \tag{1.46}$$

$$= \frac{1}{n}\sum_{i=1}^{n} y_i + \frac{(x - \overline{x})}{n\hat{\sigma}_X^2}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}) \tag{1.47}$$

$$= \frac{1}{n}\sum_{i=1}^{n} y_i + \frac{(x - \overline{x})}{n\hat{\sigma}_X^2}\sum_{i=1}^{n}(x_i - \overline{x})y_i - \frac{(x - \overline{x})}{n\hat{\sigma}_X^2}(n\overline{x} - n\overline{x})\overline{y} \tag{1.48}$$

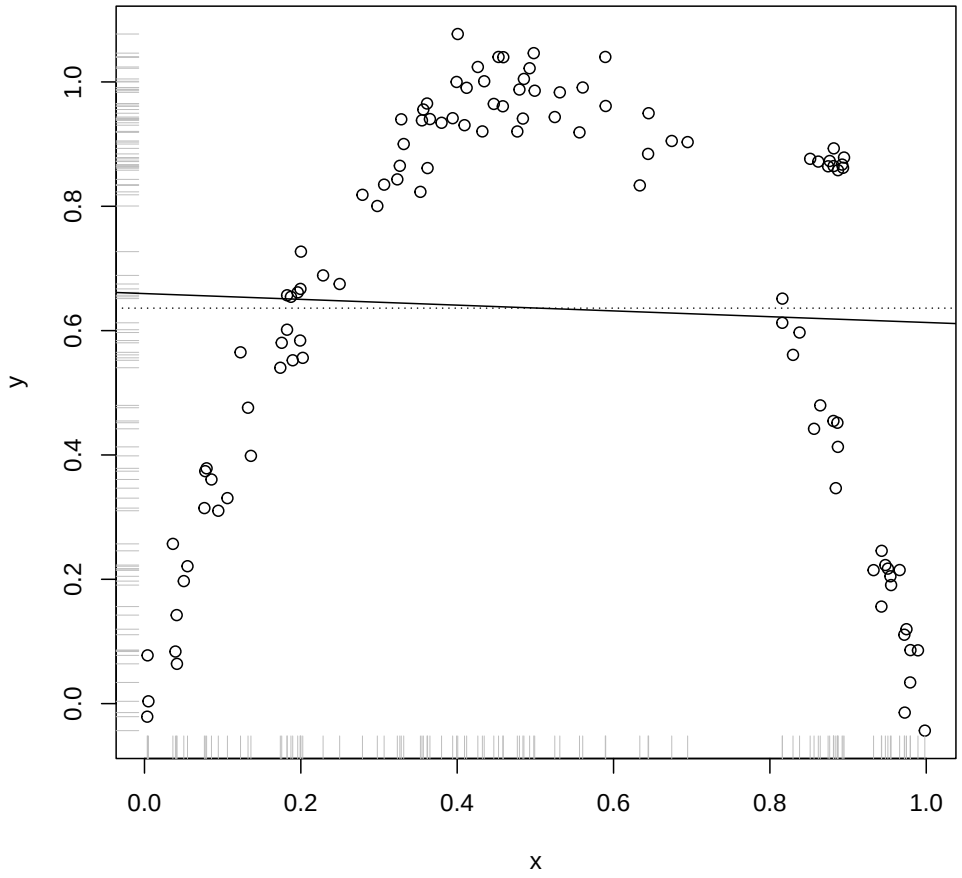
$$= \sum_{i=1}^{n}\frac{1}{n}\left(1 + \frac{(x - \overline{x})(x_i - \overline{x})}{\hat{\sigma}_X^2}\right) y_i \tag{1.49}$$

In words, our prediction is a weighted sum of the observed values y_i of the regressand, where the weights are proportional to how far x_i and x both are from the center of the data (relative to the variance of X). If x_i is on the same side of the center as x, it gets a positive weight, and if it's on the opposite side it gets a negative weight.

Figure 1.4 adds the least-squares regression line to Figure 1.1. As you can see, this is only barely slightly different from the constant regression function (the slope is X is -0.046). Visually, the problem is that there should be a positive slope in the left-hand half of the data, and a negative slope in the right, but the slopes and the densities are balanced so that the best *single* slope is near zero.[9]

Mathematically, the problem arises from the peculiar way in which least-squares linear regression smoothes the data. As I said, the weight of a data point

[9] The standard test of whether this coefficient is zero is about as far from rejecting the null hypothesis as you will ever see, $p = 0.64$. Remember this the next time you look at linear regression output.




```
plot(all.x, all.y, xlab = "x", ylab = "y")
rug(all.x, side = 1, col = "grey")
rug(all.y, side = 2, col = "grey")
abline(h = mean(all.y), lty = "dotted")
fit.all = lm(all.y ~ all.x)
abline(fit.all)
```

Figure 1.4 Data from Figure 1.1, with a horizontal line at the mean (dotted) and the ordinary least squares regression line (solid).

depends on how far it is from the *center* of the data, not how far it is from the *point at which we are trying to predict*. This works when $\mu(x)$ really is a straight line, but otherwise — e.g., here — it's a recipe for poor performance. However, it

does suggest that if we could somehow just tweak the way we smooth the data, we could do better than linear regression.

1.5 Linear Smoothers

The sample mean and the least-squares line are both special cases of **linear smoothers**, which estimates the regression function with a weighted average:

$$\widehat{\mu}(x) = \sum_i y_i \widehat{w}(x_i, x) \tag{1.50}$$

These are called linear smoothers because the predictions are linear in the *responses* y_i; as functions of x they can be and generally are nonlinear.

As I just said, the sample mean is a special case; see Exercise 1.7. Ordinary linear regression is another special case, where $\widehat{w}(x_i, x)$ is given by Eq. 1.49. Both of these, as remarked earlier, ignore how far x_i is from x. Let us look at some linear smoothers which are not so silly.

1.5.1 k-Nearest-Neighbors Regression

At the other extreme from ignoring the distance between x_i and x, we could do **nearest-neighbor regression**:

$$\widehat{w}(x_i, x) = \begin{cases} 1 & x_i \text{ nearest neighbor of } x \\ 0 & \text{otherwise} \end{cases} \tag{1.51}$$

This is very sensitive to the distance between x_i and x. If $\mu(x)$ does not change too rapidly, and X is pretty thoroughly sampled, then the nearest neighbor of x among the x_i is probably close to x, so that $\mu(x_i)$ is probably close to $\mu(x)$. However, $y_i = \mu(x_i) + \text{noise}$, so nearest-neighbor regression will include the noise into its prediction. We might instead do k-nearest-neighbors regression,

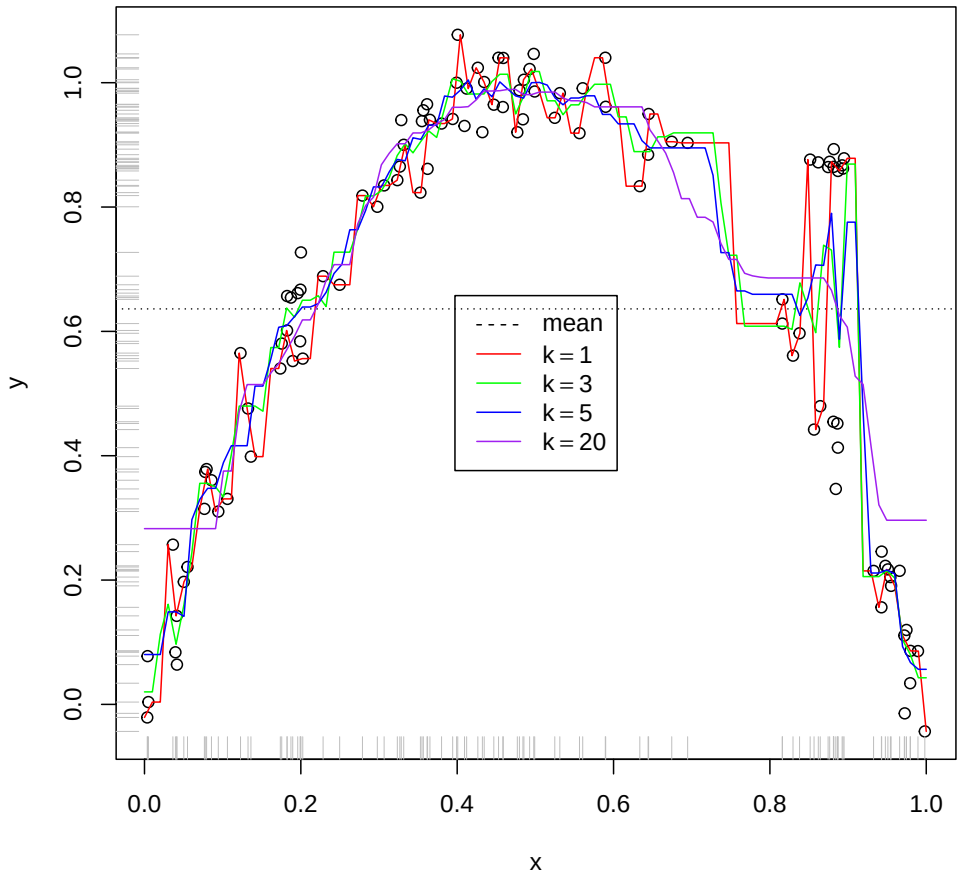
$$\widehat{w}(x_i, x) = \begin{cases} 1/k & x_i \text{ one of the } k \text{ nearest neighbors of } x \\ 0 & \text{otherwise} \end{cases} \tag{1.52}$$

Again, with enough samples all the k nearest neighbors of x are probably close to x, so their regression functions there are going to be close to the regression function at x. But because we average their values of y_i, the noise terms should tend to cancel each other out. As we increase k, we get smoother functions — in the limit $k = n$ and we just get back the constant. Figure 1.5 illustrates this for our running example data.[10] To use k-nearest-neighbors regression, we need to pick k somehow. This means we need to decide *how much* smoothing to do, and this is not trivial. We will return to this point in Chapter 3.

Because k-nearest-neighbors averages over only a fixed number of neighbors, each of which is a noisy sample, it always has some noise in its prediction, and is generally not consistent. This may not matter very much with moderately-large data (especially once we have a good way of picking k). If we want consistency,

[10] The code uses the k-nearest neighbor function provided by the package `FNN` (Beygelzimer *et al.*, 2013). This requires one to give both a set of training points (used to learn the model) and a set of test points (at which the model is to make predictions), and returns a list where the actual predictions are in the `pred` element — see `help(knn.reg)` for more, including examples.

we need to let k grow with n, but not too fast; it's enough that as $n \to \infty$, $k \to \infty$ and $k/n \to 0$ (Györfi *et al.*, 2002, Thm. 6.1, p. 88).




```
library(FNN)
plot.seq <- matrix(seq(from = 0, to = 1, length.out = 100), byrow = TRUE)
lines(plot.seq, knn.reg(train = all.x, test = plot.seq, y = all.y, k = 1)$pred, col = "red")
lines(plot.seq, knn.reg(train = all.x, test = plot.seq, y = all.y, k = 3)$pred, col = "green")
lines(plot.seq, knn.reg(train = all.x, test = plot.seq, y = all.y, k = 5)$pred, col = "blue")
lines(plot.seq, knn.reg(train = all.x, test = plot.seq, y = all.y, k = 20)$pred,
    col = "purple")
legend("center", legend = c("mean", expression(k == 1), expression(k == 3), expression(k ==
    5), expression(k == 20)), lty = c("dashed", rep("solid", 4)), col = c("black",
    "red", "green", "blue", "purple"))
```

Figure 1.5 Points from Figure 1.1 with horizontal dashed line at the mean and the k-nearest-neighbors regression curves for various k. Increasing k smooths out the regression curve, pulling it towards the mean. — The code is repetitive; can you write a function to simplify it?

1.5.2 Kernel Smoothers

Changing k in a k-nearest-neighbors regression lets us change how much smoothing we're doing on our data, but it's a bit awkward to express this in terms of a number of data points. It feels like it would be more natural to talk about a range in the independent variable over which we smooth or average. Another problem with k-NN regression is that each testing point is predicted using information from only a few of the training data points, unlike linear regression or the sample mean, which always uses all the training data. It'd be nice if we could somehow use all the training data, but in a location-sensitive way.

There are several ways to do this, as we'll see, but a particularly useful one is **kernel smoothing**, a.k.a. **kernel regression** or **Nadaraya-Watson regression**. To begin with, we need to pick a **kernel function**[11] $K(x_i, x)$ which satisfies the following properties:

1. $K(x_i, x) \geq 0$;
2. $K(x_i, x)$ depends only on the distance $x_i - x$, not the individual arguments;
3. $\int xK(0, x)dx = 0$; and
4. $0 < \int x^2 K(0, x)dx < \infty$.

These conditions together (especially the last one) imply that $K(x_i, x) \to 0$ as $|x_i - x| \to \infty$. Two examples of such functions are the density of the Unif$(-h/2, h/2)$ distribution, and the density of the standard Gaussian $\mathcal{N}(0, \sqrt{h})$ distribution. Here h can be any positive number, and is called the **bandwidth**. Because $K(x_i, x) = K(0, x_i - x)$, we will often write K as a one-argument function, $K(x_i - x)$. Because we often want to consider similar kernels which differ only by bandwidth, we'll either write $K(\frac{x_i - x}{h})$, or $K_h(x_i - x)$.

The Nadaraya-Watson estimate of the regression function is

$$\widehat{\mu}(x) = \sum_i y_i \frac{K(x_i, x)}{\sum_j K(x_j, x)} \tag{1.53}$$

i.e., in terms of Eq. 1.50,

$$\widehat{w}(x_i, x) = \frac{K(x_i, x)}{\sum_j K(x_j, x)} \tag{1.54}$$

(Notice that here, as in k-NN regression, the sum of the weights is always 1. Why?)[12]

What does this achieve? Well, $K(x_i, x)$ is large if x_i is close to x, so this will place a lot of weight on the training data points close to the point where we are trying to predict. More distant training points will have smaller weights, falling

[11] There are many other mathematical objects which are *also* called "kernels". Some of these meanings are related, but not all of them. (Cf. "normal".)

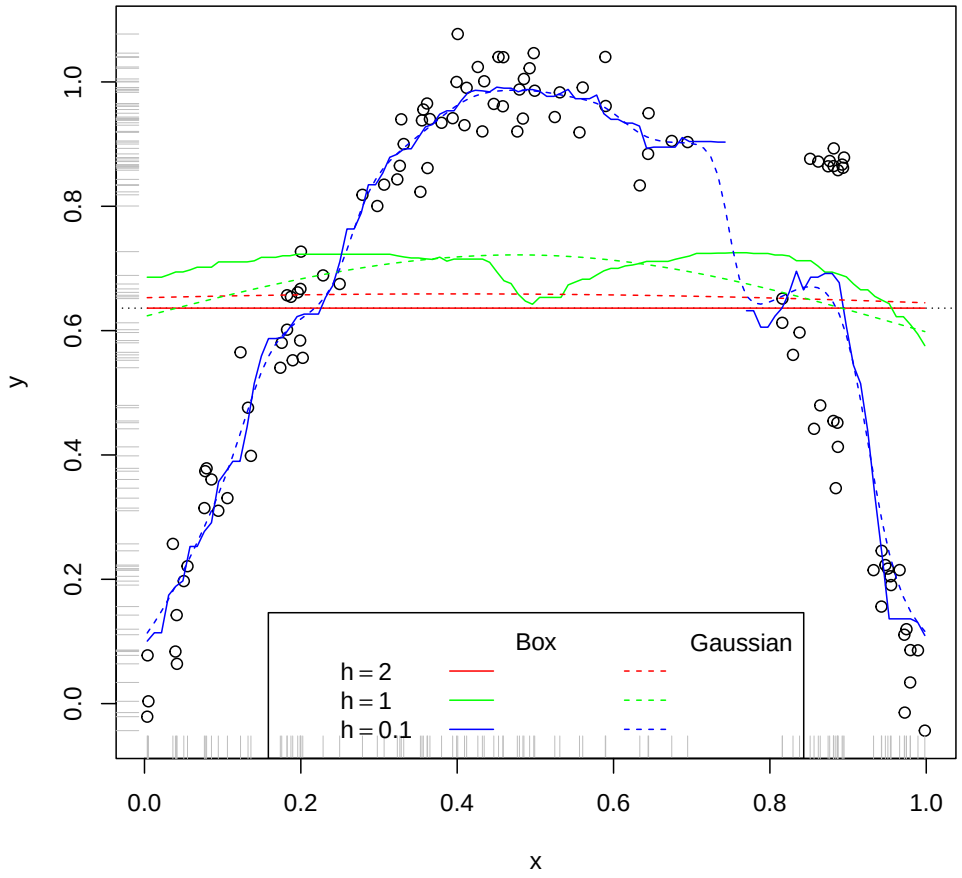
[12] What do we do if $K(x_i, x)$ is zero for some x_i? Nothing; they just get zero weight in the average. What do we do if *all* the $K(x_i, x)$ are zero? Different people adopt different conventions; popular ones are to return the global, unweighted mean of the y_i, to do some sort of interpolation from regions where the weights are defined, and to throw up our hands and refuse to make any predictions (computationally, return `NA`).

off towards zero. If we try to predict at a point x which is very far from any of the training data points, the value of $K(x_i, x)$ will be small for all x_i, but it will typically be *much, much smaller* for all the x_i which are not the nearest neighbor of x, so $\widehat{w}(x_i, x) \approx 1$ for the nearest neighbor and ≈ 0 for all the others.[13] That is, far from the training data, our predictions will tend towards nearest neighbors, rather than going off to $\pm\infty$, as linear regression's predictions do. Whether this is good or bad of course depends on the true $\mu(x)$ — and how often we have to predict what will happen very far from the training data.

Figure 1.6 shows our running example data, together with kernel regression estimates formed by combining the uniform-density, or **box**, and Gaussian kernels with different bandwidths. The box kernel simply takes a region of width h around the point x and averages the training data points it finds there. The Gaussian kernel gives reasonably large weights to points within h of x, smaller ones to points within $2h$, tiny ones to points within $3h$, and so on, shrinking like $e^{-(x-x_i)^2/2h}$. As promised, the bandwidth h controls the degree of smoothing. As $h \to \infty$, we revert to taking the global mean. As $h \to 0$, we tend to get spikier functions — with the Gaussian kernel at least it tends towards the nearest-neighbor regression.

If we want to use kernel regression, we need to choose both which kernel to use, and the bandwidth to use with it. Experience, like Figure 1.6, suggests that the bandwidth usually matters a lot more than the kernel. This puts us back to roughly where we were with k-NN regression, needing to control the degree of smoothing, without knowing how smooth $\mu(x)$ really is. Similarly again, with a fixed bandwidth h, kernel regression is generally not consistent. However, if $h \to 0$ as $n \to \infty$, but doesn't shrink *too* fast, then we can get consistency.

[13] Take a Gaussian kernel in one dimension, for instance, so $K(x_i, x) \propto e^{-(x_i-x)^2/2h^2}$. Say x_i is the nearest neighbor, and $|x_i - x| = L$, with $L \gg h$. So $K(x_i, x) \propto e^{-L^2/2h^2}$, a small number. But now for any other x_j, $K(x_i, x) \propto e^{-L^2/2h^2} e^{-(x_j-x_i)L/2h^2} e^{-(x_j-x_i)^2/2h^2} \ll e^{-L^2/2h^2}$. — This assumes that we're using a kernel like the Gaussian, which never quite goes to zero, unlike the box kernel.




```
lines(ksmooth(all.x, all.y, "box", bandwidth = 2), col = "red")
lines(ksmooth(all.x, all.y, "box", bandwidth = 1), col = "green")
lines(ksmooth(all.x, all.y, "box", bandwidth = 0.1), col = "blue")
lines(ksmooth(all.x, all.y, "normal", bandwidth = 2), col = "red", lty = "dashed")
lines(ksmooth(all.x, all.y, "normal", bandwidth = 1), col = "green", lty = "dashed")
lines(ksmooth(all.x, all.y, "normal", bandwidth = 0.1), col = "blue", lty = "dashed")
legend("bottom", ncol = 3, legend = c("", expression(h == 2), expression(h == 1),
    expression(h == 0.1), "Box", "", "", "", "Gaussian", "", "", ""), lty = c("blank",
    "blank", "blank", "blank", "blank", "blank", "solid", "solid", "solid", "blank", "dashed",
    "dashed", "dashed"), col = c("black", "black", "black", "black", "black", "red",
    "green", "blue", "black", "red", "green", "blue"), pch = NA)
```

Figure 1.6 Data from Figure 1.1 together with kernel regression lines, for various combinations of kernel (box/uniform or Gaussian) and bandwidth. Note the abrupt jump around $x = 0.75$ in the $h = 0.1$ box-kernel (solid blue) line — with a small bandwidth the box kernel is unable to interpolate smoothly across the break in the training data, while the Gaussian kernel (dashed blue) can.

1.5.3 Some General Theory for Linear Smoothers

Some key parts of the theory you are familiar with for linear regression models carries over more generally to linear smoothers. They are not quite so important any more, but they do have their uses, and they can serve as security objects during the transition to non-parametric regression.

Throughout this sub-section, we will temporarily assume that $Y = \mu(X) + \epsilon$, with the noise terms ϵ having constant variance σ^2, no correlation with the noise at other observations. Also, we will define the **smoothing**, **influence** or **hat matrix** $\hat{\mathbf{w}}$ by $\hat{w}_{ij} = \hat{w}(x_i, x_j)$. This records how much influence observation y_j had on the smoother's fitted value for $\mu(x_i)$, which (remember) is $\widehat{\mu}(x_i)$ or $\widehat{\mu}_i$ for short[14], hence the name "hat matrix" for $\hat{w}$.

1.5.3.1 Standard error of predicted mean values

It is easy to get the standard error of any predicted mean value $\widehat{\mu}(x)$, by first working out its variance:

$$\mathbb{V}\left[\widehat{\mu}(x)\right] = \mathbb{V}\left[\sum_{j=1}^{n} w(x_j, x) Y_j\right] \tag{1.55}$$

$$= \sum_{j=1}^{n} \mathbb{V}\left[w(x_j, x) Y_j\right] \tag{1.56}$$

$$= \sum_{j=1}^{n} w^2(x_j, x) \mathbb{V}\left[Y_j\right] \tag{1.57}$$

$$= \sigma^2 \sum_{j=1}^{n} w^2(x_j, x) \tag{1.58}$$

The second line uses the assumption that the noise is uncorrelated, and the last the assumption that the noise variance is constant. In particular, for a point x_i which appeared in the training data, $\mathbb{V}\left[\widehat{\mu}(x_i)\right] = \sigma^2 \sum_j w_{ij}^2$.

Notice that this is the variance in the predicted *mean* value, $\widehat{\mu}(x)$. It is not an estimate of $\mathbb{V}\left[Y \mid X = x\right]$, though we will see how conditional variances can be estimated using nonparametric regression in Chapter 10.

Notice also that we have *not* had to assume that the noise is Gaussian. If we did add that assumption, this formula would also give us a confidence interval for the fitted value (though we would still have to worry about estimating σ).

1.5.3.2 (Effective) Degrees of Freedom

For linear regression models, you will recall that the number of "degrees of freedom" was just the number of coefficients (including the intercept). While degrees of freedom are less important for other sorts of regression than for linear models, they're still worth knowing about, so I'll explain here how they are defined and

[14] This is often written as $\hat{y}_i$, but that's not very logical notation; the quantity is a *function* of y_i, not an estimate of it; it's an *estimate* of $\mu(x_i)$.

calculated. In general, we can't use the number of parameters to define degrees of freedom, since most linear smoothers don't *have* parameters. Instead, we have to go back to the reasons *why* the number of parameters actually matters in ordinary linear models. (Linear algebra follows.)

We'll start with an $n \times p$ data matrix of predictor variables $\mathbf{x}$ (possibly including an all-1 column for an intercept), and an $n \times 1$ column matrix of response values $\mathbf{y}$. The ordinary least squares estimate of the p-dimensional coefficient vector β is

$$\hat{\beta} = \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\mathbf{y} \tag{1.59}$$

This lets us write the fitted values in terms of $\mathbf{x}$ and $\mathbf{y}$ alone:

$$\widehat{\mu} = \mathbf{x}\hat{\beta} \tag{1.60}$$

$$= \left(\mathbf{x}\left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\right)\mathbf{y} \tag{1.61}$$

$$= \mathbf{w}\mathbf{y} \tag{1.62}$$

where $\mathbf{w}$ is the $n \times n$ matrix, with w_{ij} saying how much of each observed y_j contributes to each fitted $\widehat{\mu}_i$. This is what, a little while ago, I called the influence or hat matrix, in the special case of ordinary least squares.

Notice that $\mathbf{w}$ depends *only* on the predictor variables in $\mathbf{x}$; the observed response values in $\mathbf{y}$ don't matter. If $\mathbf{y}$ changes, the fitted values $\widehat{\mu}$ will also change, but only within the limits allowed by $\mathbf{w}$. There are n independent coordinates along which $\mathbf{y}$ can change, so we say the data have n degrees of freedom. Once $\mathbf{x}$ (and thus $\mathbf{w}$) are fixed, however, $\widehat{\mu}$ has to lie in a p-dimensional linear subspace in this n-dimensional space, and the residuals have to lie in the $(n-p)$-dimensional space orthogonal to it.

Geometrically, the dimension of the space in which $\widehat{\mu} = \mathbf{w}\mathbf{y}$ is confined is the rank of the matrix $\mathbf{w}$. Since $\mathbf{w}$ is an idempotent matrix (Exercise 1.5), its rank equals its trace. And that trace is, exactly, p:

$$\mathrm{tr}\,\mathbf{w} = \mathrm{tr}\left(\mathbf{x}\left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\right) \tag{1.63}$$

$$= \mathrm{tr}\left(\mathbf{x}^T\mathbf{x}\left(\mathbf{x}^T\mathbf{x}\right)^{-1}\right) \tag{1.64}$$

$$= \mathrm{tr}\,\mathbf{I}_p = p \tag{1.65}$$

since for any matrices $\mathbf{a}, \mathbf{b}$, $\mathrm{tr}\,(\mathbf{ab}) = \mathrm{tr}\,(\mathbf{ba})$, and $\mathbf{x}^T\mathbf{x}$ is a $p \times p$ matrix[15].

For more general linear smoothers, we can still write Eq. 1.50 in matrix form,

$$\widehat{\mu} = \mathbf{w}\mathbf{y} \tag{1.66}$$

We now *define* the degrees of freedom[16] to be the trace of $\mathbf{w}$:

$$df(\widehat{\mu}) \equiv \mathrm{tr}\,\mathbf{w} \tag{1.67}$$

This may not be an integer.

[15] This all assumes that $\mathbf{x}^T\mathbf{x}$ has an inverse. Can you work out what happens when it does not?

[16] Some authors prefer to say "*effective* degrees of freedom", to emphasize that we're not just counting parameters.

1.5.3.3 Covariance of Observations and Fits

Eq. 1.67 defines the number of degrees of freedom for linear smoothers. A yet more general definition includes nonlinear methods, assuming that $Y_i = \mu(x_i) + \epsilon_i$, and the ϵ_i consist of uncorrelated noise of constant[17] variance σ^2. This is

$$gdf(\widehat{\mu}) \equiv \frac{1}{\sigma^2}\sum_{i=1}^{n}\mathrm{Cov}\left[Y_i, \widehat{\mu}(x_i)\right] \tag{1.68}$$

In words, this is the normalized covariance between each observed response Y_i and the corresponding predicted value, $\widehat{\mu}(x_i)$. This is a very natural way of measuring how flexible or stable the regression model is, by seeing how much it shifts with the data.

If we do have a linear smoother, Eq. 1.68 reduces to Eq. 1.67.

$$\mathrm{Cov}\left[Y_i, \widehat{\mu}(x_i)\right] = \mathrm{Cov}\left[Y_i, \sum_{j=1}^{n} w_{ij}Y_j\right] \tag{1.69}$$

$$= \sum_{j=1}^{n} w_{ij}\mathrm{Cov}\left[Y_i, Y_j\right] \tag{1.70}$$

$$= w_{ii}\mathbb{V}\left[Y_i\right] = \sigma^2 w_{ii} \tag{1.71}$$

$$\tag{1.72}$$

Here the first line uses the fact that we're dealing with a linear smoother, and the last line the assumption that ϵ_i is uncorrelated and has constant variance. Therefore

$$gdf(\widehat{\mu}) = \frac{1}{\sigma^2}\sum_{i=1}^{n}\sigma^2 w_{ii} = \mathrm{tr}\,\mathbf{w} = df(\widehat{\mu}) \tag{1.73}$$

as promised.

1.5.3.4 Prediction Errors

Bias

Because linear smoothers are linear in the response variable, it's easy to work out (theoretically) the expected value of their fits:

$$\mathbb{E}\left[\widehat{\mu}_i\right] = \sum_{j=1}^{n} w_{ij}\mathbb{E}\left[Y_j\right] \tag{1.74}$$

In matrix form,

$$\mathbb{E}\left[\widehat{\mu}\right] = \mathbf{w}\mathbb{E}\left[\mathbf{Y}\right] \tag{1.75}$$

This means the smoother is unbiased if, and only if, $\mathbf{w}\mathbb{E}\left[\mathbf{Y}\right] = \mathbb{E}\left[\mathbf{Y}\right]$, that is, if $\mathbb{E}\left[\mathbf{Y}\right]$ is an eigenvector of $\mathbf{w}$. Turned around, the condition for the smoother to be unbiased is

$$(\mathbf{I}_n - \mathbf{w})\mathbb{E}\left[\mathbf{Y}\right] = \mathbf{0} \tag{1.76}$$

[17] But see Exercise 1.10.

In general, $(\mathbf{I}_n - \mathbf{w})\mathbb{E}[\mathbf{Y}] \neq \mathbf{0}$, so linear smoothers are more or less biased. Different smoothers are, however, unbiased for different families of regression functions. Ordinary linear regression, for example, is unbiased if and only if the regression function really is linear.

In-sample mean squared error

When you studied linear regression, you learned that the expected mean-squared error on the data used to fit the model is $\sigma^2(n-p)/n$. This formula generalizes to other linear smoothers. Let's first write the residuals in matrix form.

$$\mathbf{y} - \widehat{\mu} = \mathbf{y} - \mathbf{w}\mathbf{y} \tag{1.77}$$
$$= \mathbf{I}_n\mathbf{y} - \mathbf{w}\mathbf{y} \tag{1.78}$$
$$= (\mathbf{I}_n - \mathbf{w})\mathbf{y} \tag{1.79}$$

The in-sample mean squared error is $n^{-1}\left\|\mathbf{y} - \widehat{\mu}\right\|^2$, so

$$\frac{1}{n}\left\|\mathbf{y} - \widehat{\mu}\right\|^2 = \frac{1}{n}\left\|(\mathbf{I}_n - \mathbf{w})\mathbf{y}\right\|^2 \tag{1.80}$$
$$= \frac{1}{n}\mathbf{y}^T(\mathbf{I}_n - \mathbf{w}^T)(\mathbf{I}_n - \mathbf{w})\mathbf{y} \tag{1.81}$$

Taking expectations[18],

$$\mathbb{E}\left[\frac{1}{n}\left\|\mathbf{y} - \widehat{\mu}\right\|^2\right] = \frac{\sigma^2}{n}\operatorname{tr}\left((\mathbf{I}_n - \mathbf{w}^T)(\mathbf{I}_n - \mathbf{w})\right) + \frac{1}{n}\left\|(\mathbf{I}_n - \mathbf{w})\mathbb{E}[\mathbf{y}]\right\|^2 \tag{1.82}$$
$$= \frac{\sigma^2}{n}\left(\operatorname{tr}\mathbf{I}_n - 2\operatorname{tr}\mathbf{w} + \operatorname{tr}(\mathbf{w}^T\mathbf{w})\right) + \frac{1}{n}\left\|(\mathbf{I}_n - \mathbf{w})\mathbb{E}[\mathbf{y}]\right\|^2 \tag{1.83}$$
$$= \frac{\sigma^2}{n}\left(n - 2\operatorname{tr}\mathbf{w} + \operatorname{tr}(\mathbf{w}^T\mathbf{w})\right) + \frac{1}{n}\left\|(\mathbf{I}_n - \mathbf{w})\mathbb{E}[\mathbf{y}]\right\|^2 \tag{1.84}$$

The last term, $n^{-1}\left\|(\mathbf{I}_n - \mathbf{w})\mathbb{E}[\mathbf{y}]\right\|^2$, comes from the bias: it indicates the distortion that the smoother would impose on the regression function, even without noise. The first term, proportional to σ^2, reflects the variance. Notice that it involves not only what we've called the degrees of freedom, $\operatorname{tr}\mathbf{w}$, but also a second-order term, $\operatorname{tr}\mathbf{w}^T\mathbf{w}$. For ordinary linear regression, you can show (Exercise 1.9) that $\operatorname{tr}(\mathbf{w}^T\mathbf{w}) = p$, so $2\operatorname{tr}\mathbf{w} - \operatorname{tr}(\mathbf{w}^T\mathbf{w})$ would also equal p. For this reason, some people prefer either $\operatorname{tr}(\mathbf{w}^T\mathbf{w})$ or $2\operatorname{tr}\mathbf{w} - \operatorname{tr}(\mathbf{w}^T\mathbf{w})$ as the definition of degrees of freedom for linear smoothers, so be careful.

1.5.3.5 Inferential Statistics

Many of the formulas underlying things like the F test (for whether a regression predicts significantly better than the global mean) carry over from linear regression to linear smoothers, if one uses the right definitions of degrees of freedom, *and* one believes that the noise is always IID and Gaussian. However, we will

[18] By using the general result that $\mathbb{E}\left[\vec{X}\cdot\mathbf{a}\vec{X}\right] = \operatorname{tr}(\mathbf{a}\mathbb{V}\left[\vec{X}\right]) + \mathbb{E}\left[\vec{X}\right]\cdot\mathbf{a}\mathbb{E}\left[\vec{X}\right]$ for any random vector $\vec{X}$ and non-random square matrix $\mathbf{a}$.

see ways of doing inference on regression models which don't rely on Gaussian assumptions at all (Ch. 6), so I won't go over these results.

1.6 Further Reading

In Chapter 2, we'll look more at the limits of linear regression and some extensions; Chapter 3 will cover some key aspects of evaluating statistical models, including regression models; and then Chapter 4 will come back to kernel regression, and more powerful tools than `ksmooth`. Chapters 10–8 and 13 all introduce further regression methods, while Chapters 11–12 pursue extensions.

Good treatments of regression, emphasizing linear smoothers but *not* limited to linear regression, can be found in Wasserman (2003, 2006), Simonoff (1996), Faraway (2006) and Györfi *et al.* (2002). The last of these in particular provides a very thorough theoretical treatment of non-parametric regression methods.

On generalizations of degrees of freedom to non-linear models, see Buja *et al.* (1989, §2.7.3), and Ye (1998).

Historical notes

All the forms of nonparametric regression covered in this chapter are actually quite old. Kernel regression was introduced independently by Nadaraya (1964) and Watson (1964). The origin of nearest neighbor methods is less clear, and indeed they may have been independently invented multiple times[19] — Cover and Hart (1967) collects some of the relevant early citations, as well as providing a pioneering theoretical analysis, extended to regression problems in Cover (1968a,b).

Defining the effective degrees of freedom of a linear smoother to be $\mathrm{tr}\,\mathbf{w}$ is usually attributed to Hastie and Tibshirani (1990). Results showing that many equations for linear regression models generalized to linear smoothers, with $\mathrm{tr}\,w$ in the role of the number of coefficients, were derived in more or less generality by a number of authors in the 1980s (Efron, 1986; Wahba, 1990). Defining the average covariance between fitted values and observed values as the number of generalized degrees of freedom of non-linear smoothers seems to have first been explicit in Ye (1998), though, again, the use of such covariances is older, as reviewed by Efron (2004).

Exercises

1.1 Suppose $Y_1, Y_2, \ldots Y_n$ are random variables with the same mean μ and standard deviation σ, and that they are all uncorrelated with each other, but not necessarily independent[20] or identically distributed. Show the following:

[19] Pelillo (2014) claims that the oldest clear expression of the idea of a nearest-neighbor classification rule is found in the treatise known as *The Book of Optics* (c. 1030) by the medieval Islamic scientist Abu Ali al-Hasan ibn al-Hasan ibn al-Haytham, known in Europe, from Latin translations of his work, as "Alhazen". This seems very plausible to me, based on the quotations provided in that paper, but I don't know what actual historians of science make of the argument.

[20] See Appendix **??** for a refresher on the difference between "uncorrelated" and "independent".

1. $\mathbb{V}\left[\sum_{i=1}^{n} Y_i\right] = n\sigma^2$.
2. $\mathbb{V}\left[n^{-1}\sum_{i=1}^{n} Y_i\right] = \sigma^2/n$.
3. The standard deviation of $n^{-1}\sum_{i=1}^{n} Y_i$ is $\sigma/\sqrt{n}$.
4. The standard deviation of $\mu - n^{-1}\sum_{i=1}^{n} Y_i$ is $\sigma/\sqrt{n}$.

Can you state the analogous results when the Y_i share mean μ but each has its own standard deviation σ_i? When each Y_i has a distinct mean μ_i? (Assume in both cases that the Y_i remain uncorrelated.)

1.2 Suppose we use the **mean absolute error** instead of the mean squared error:

$$\mathrm{MAE}(m) = \mathbb{E}\left[|Y - m|\right] \tag{1.85}$$

Is this also minimized by taking $m = \mathbb{E}[Y]$? If not, what value $\tilde{\mu}$ minimizes the MAE? Should we use MSE or MAE to measure error?

1.3 Derive Eqs. 1.42 and 1.41 by minimizing Eq. 1.40.

1.4 What does it mean to say that Gaussian kernel regression approaches nearest-neighbor regression as $h \to 0$? Why does it do so? Is this true for all kinds of kernel regression?

1.5 Prove that $\mathbf{w}$ from Eq. 1.62 is idempotent, i.e., that $\mathbf{w}^2 = \mathbf{w}$.

1.6 Show that for ordinary linear regression, Eq. 1.58 gives the same variance for fitted values as the usual formula.

1.7 Consider the global mean as a linear smoother. Work out the influence matrix $\mathbf{w}$, and show that it has one degree of freedom, using the definition in Eq. 1.67.

1.8 Consider k-nearest-neighbors regression as a linear smoother. Work out the influence matrix $\mathbf{w}$, and find an expression for the number of degrees of freedom (in the sense of Eq. 1.67) in terms of k and n. *Hint:* Your answers should reduce to those of the previous problem when $k = n$.

1.9 Suppose that $Y_i = \mu(x_i) + \epsilon_i$, where the ϵ_i are uncorrelated have mean 0, with constant variance σ^2. Prove that, for a linear smoother, $n^{-1}\sum_{i=1}^{n} \mathbb{V}[\hat{\mu}_i] = (\sigma^2/n)\,\mathrm{tr}\,(\mathbf{w}\mathbf{w}^T)$. Show that this reduces to $\sigma^2 p/n$ for ordinary linear regression.

1.10 Suppose that $Y_i = \mu(x_i) + \epsilon_i$, where the ϵ_i are uncorrelated and have mean 0, but each has its own variance σ_i^2. Consider modifying the definition of degrees of freedom to $\sum_{i=1}^{n} \mathrm{Cov}\left[Y_i, \hat{\mu}_i\right]/\sigma_i^2$ (which reduces to Eq. 1.68 if all the $\sigma_i^2 = \sigma^2$). Show that this still equals $\mathrm{tr}\,\mathbf{w}$ for a linear smoother with influence matrix $\mathbf{w}$.

2

The Truth about Linear Regression

We need to say some more about how linear regression, and especially about how it really works and how it can fail. Linear regression is important because

1. it's a fairly straightforward technique which sometimes works tolerably for prediction;
2. it's a simple foundation for some more sophisticated techniques;
3. it's a standard method so people use it to communicate; and
4. it's a standard method so people have come to confuse it with prediction and even with causal inference as such.

We need to go over (1)–(3), and provide prophylaxis against (4).

2.1 Optimal Linear Prediction: Multiple Variables

We have a numerical variable Y and a p-dimensional vector of predictor variables or features $\vec{X}$. We would like to predict Y using $\vec{X}$. Chapter 1 taught us that the mean-squared optimal predictor is is the conditional expectation,

$$\mu(\vec{x}) = \mathbb{E}\left[Y \mid \vec{X} = \vec{x}\right] \tag{2.1}$$

Instead of using the optimal predictor $\mu(\vec{x})$, let's try to predict as well as possible while using only a linear[1] function of $\vec{x}$, say $\beta_0 + \beta \cdot \vec{x}$. This is not an assumption about the world, but rather a decision on our part; a choice, not a hypothesis. This decision can be good — $\beta_0 + \vec{x} \cdot \beta$ could be a tolerable approximation to $\mu(\vec{x})$ — even if the linear hypothesis is strictly wrong. Even if no linear approximation to μ is much good mathematically, but we might still want one for practical reasons, e.g., speed of computation.

(Perhaps the best reason to hope the choice to use a linear model isn't crazy is that we may hope μ is a smooth function. If it is, then we can Taylor expand[2] it about our favorite point, say $\vec{u}$:

$$\mu(\vec{x}) = \mu(\vec{u}) + \sum_{i=1}^{p} \left(\left. \frac{\partial \mu}{\partial x_i} \right|_{\vec{u}} \right) (x_i - u_i) + O(\|\vec{x} - \vec{u}\|^2) \tag{2.2}$$

[1] Pedants might quibble that this function is actually affine rather than linear. But the distinction is specious: we can always add an extra element to $\vec{x}$, which is always 1, getting the vector $\vec{x}'$, and then we have the linear function $\beta' \cdot \vec{x}'$.

[2] See Appendix B on Taylor approximations.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

or, in the more compact vector-calculus notation,

$$\mu(\vec{x}) = \mu(\vec{u}) + (\vec{x} - \vec{u}) \cdot \nabla\mu(\vec{u}) + O(\|\vec{x} - \vec{u}\|^2) \tag{2.3}$$

If we only look at points $\vec{x}$ which are close to $\vec{u}$, then the remainder terms $O(\|\vec{x} - \vec{u}\|^2)$ are small, and a linear approximation is a good one[3]. Here, "close to $\vec{u}$" really means "so close that all the non-linear terms in the Taylor series are comparatively negligible".)

Whatever the reason for wanting to use a linear function, there are many linear functions, and we need to pick just one of them. We may as well do that by minimizing mean-squared error again:

$$MSE(\beta) = \mathbb{E}\left[\left(Y - \beta_0 - \vec{X} \cdot \beta\right)^2\right] \tag{2.4}$$

Going through the optimization is parallel to the one-dimensional case we worked through in §1.4.3, with the conclusions that

$$\beta_0 = \mathbb{E}[Y] - \beta \cdot \mathbb{E}\left[\vec{X}\right] \tag{2.5}$$

just as in the one-dimensional case (Exercise 2.1); and that the optimal β is

$$\beta = \mathbf{v}^{-1}\mathrm{Cov}\left[\vec{X}, Y\right] \tag{2.6}$$

where $\mathbf{v}$ is the covariance matrix of $\vec{X}$, i.e., $v_{ij} = \mathrm{Cov}[X_i, X_j]$, and $\mathrm{Cov}\left[\vec{X}, Y\right]$ is the vector of covariances between the regressors and Y, i.e. $\mathrm{Cov}\left[\vec{X}, Y\right]_i = \mathrm{Cov}[X_i, Y]$. Another way to view this is that the optimal linear prediction function $\lambda(\vec{x})$ is always

$$\lambda(\vec{x}) = \mathbb{E}[Y] + (\vec{x} - \mathbb{E}\left[\vec{X}\right])\mathbf{v}^{-1}\mathrm{Cov}\left[\vec{X}, Y\right] \tag{2.7}$$

These conclusions hold without assuming anything at all about the true regression function μ; about the distribution of X, of Y, of $Y \mid X$, or of $Y - \mu(X)$ (in particular, nothing needs to be Gaussian); or whether data points are independent or not.

Multiple regression would be a lot simpler if we could just do a simple regression for each regressor, and add them up; but really, this is what multiple regression *does*, just in a disguised form. If the input variables are uncorrelated, $\mathbf{v}$ is diagonal ($v_{ij} = 0$ unless $i = j$), and so is $\mathbf{v}^{-1}$. Then doing multiple regression breaks up into a sum of separate simple regressions across each input variable. When the input variables are correlated and $\mathbf{v}$ is not diagonal, we can think of the multiplication by $\mathbf{v}^{-1}$ as **de-correlating** $\vec{X}$ — applying a linear transformation to come up with a new set of inputs which are uncorrelated with each other.[4]

Notice: β depends on the marginal distribution of $\vec{X}$ (through the covariance

[3] If you are not familiar with the big-O notation like $O(\|\vec{x} - \vec{u}\|^2)$, now would be a good time to read Appendix A.

[4] If $\vec{Z}$ is a random vector with covariance matrix I, then $\mathbf{w}\vec{Z}$ is a random vector with covariance matrix $\mathbf{w}^T\mathbf{w}$. Conversely, if we start with a random vector $\vec{X}$ with covariance matrix $\mathbf{v}$, the latter

matrix $\mathbf{v}$). If that shifts, the optimal coefficients β will shift, *unless* the real regression function is linear.

2.1.1 Collinearity

The formula $\beta = \mathbf{v}^{-1}\text{Cov}\left[\vec{X}, Y\right]$ makes no sense if $\mathbf{v}$ has no inverse. This will happen if, and only if, the predictor variables are linearly dependent on each other — if one of the predictors is really a linear combination of the others. Then (as we learned in linear algebra) the covariance matrix is of less than "full rank" (i.e., "rank deficient") and it doesn't have an inverse. Equivalently, $\mathbf{v}$ has at least one eigenvalue which is exactly zero.

So much for the algebra; what does that mean statistically? Let's take an easy case where one of the predictors is just a multiple of the others — say you've included people's weight in pounds (X_1) and mass in kilograms (X_2), so $X_1 = 2.2X_2$. Then if we try to predict Y, we'd have

$$\widehat{\mu}(\vec{X}) = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \ldots + \beta_p X_p \tag{2.8}$$

$$= 0X_1 + (2.2\beta_1 + \beta_2)X_2 + \sum_{i=3}^{p} \beta_i X_i \tag{2.9}$$

$$= (\beta_1 + \beta_2/2.2)X_1 + 0X_2 + \sum_{i=3}^{p} \beta_i X_i \tag{2.10}$$

$$= -2200X_1 + (1000 + \beta_1 + \beta_2)X_2 + \sum_{i=3}^{p} \beta_i X_i \tag{2.11}$$

In other words, because there's a linear relationship between X_1 and X_2, we make the coefficient for X_1 whatever we like, provided we adjust the coefficient for X_2 to compensate, and it has no effect at all on our prediction. So rather than having one optimal linear predictor, we have infinitely many of them.[5]

There are three ways of dealing with collinearity. One is to get a different data set where the regressors are no longer collinear. A second is to identify one of the collinear variables (it usually doesn't matter which) and drop it from the data set. This can get complicated; principal components analysis (Chapter 15) can help here. Thirdly, since the issue is that there are infinitely many different coefficient vectors which all minimize the MSE, we could appeal to some extra principle, beyond prediction accuracy, to select just one of them. We might, for instance,

has a "square root" $\mathbf{v}^{1/2}$ (i.e., $\mathbf{v}^{1/2}\mathbf{v}^{1/2} = \mathbf{v}$), and $\mathbf{v}^{-1/2}\vec{X}$ will be a random vector with covariance matrix $\mathbf{I}$. When we write our predictions as $\vec{x}\mathbf{v}^{-1}\text{Cov}\left[\vec{X}, Y\right]$, we should think of this as $\left(\vec{x}\mathbf{v}^{-1/2}\right)\left(\mathbf{v}^{-1/2}\text{Cov}\left[\vec{X}, Y\right]\right)$. We use one power of $\mathbf{v}^{-1/2}$ to transform the input features into uncorrelated variables before taking their correlations with the response, and the other power to decorrelate $\vec{x}$. — For more on using covariance matrices to come up with new, decorrelated variables, see Chapter 15.

[5] Algebraically, there is a linear combination of two (or more) of the regressors which is constant. The coefficients of this linear combination are given by one of the zero eigenvectors of $\mathbf{v}$.

prefer smaller coefficient vectors (all else being equal), or ones where more of the coefficients were exactly zero. Using some quality other than the squared error to pick out a unique solution is called "regularizing" the optimization problem, and a lot of attention has been given to regularized regression, especially in the "high dimensional" setting where the number of coefficients is comparable to, or even greater than, the number of data points. See Appendix D.3.5, and exercise 7.2 in Chapter 7.

2.1.2 The Prediction and Its Error

Once we have coefficients β, we can use them to make predictions for the expected value of Y at *arbitrary* values of $\vec{X}$, whether we've an observation there before or not. How good are these?

If we have the optimal coefficients, then the prediction error will be uncorrelated with the regressors:

$$\begin{aligned}
\text{Cov}\left[Y - \vec{X}\cdot\beta, \vec{X}\right] &= \text{Cov}\left[Y, \vec{X}\right] - \text{Cov}\left[\vec{X}\cdot(\mathbf{v}^{-1}\text{Cov}\left[\vec{X}, Y\right]), \vec{X}\right] && (2.12)\\
&= \text{Cov}\left[Y, \vec{X}\right] - \mathbf{v}\mathbf{v}^{-1}\text{Cov}\left[Y, \vec{X}\right] && (2.13)\\
&= 0 && (2.14)
\end{aligned}$$

Moreover, the expected prediction error, averaged over all $\vec{X}$, will be zero (Exercise 2.2):

$$\mathbb{E}\left[Y - \vec{X}\cdot\beta\right] = 0 \tag{2.15}$$

But the conditional expectation of the error is generally not zero,

$$\mathbb{E}\left[Y - \vec{X}\cdot\beta \mid \vec{X} = \vec{x}\right] \neq 0 \tag{2.16}$$

and the conditional variance is generally not constant,

$$\mathbb{V}\left[Y - \vec{X}\cdot\beta \mid \vec{X} = \vec{x}_1\right] \neq \mathbb{V}\left[Y - \vec{X}\cdot\beta \mid \vec{X} = \vec{x}_2\right] \tag{2.17}$$

The optimal linear predictor can be arbitrarily bad, and it can make arbitrarily big systematic mistakes. It is generally very biased[6].

2.1.3 Estimating the Optimal Linear Predictor

To actually estimate β from data, we need to make some probabilistic assumptions about where the data comes from. A fairly weak but often sufficient assumption is that observations $(\vec{X}_i, Y_i)$ are independent for different values of i, with unchanging covariances. Then if we look at the sample covariances, they will, by

[6] You were taught in your linear models course that linear regression makes unbiased predictions. This presumed that the linear model was true.

the law of large numbers, converge on the true covariances:

$$\frac{1}{n}\mathbf{X}^T\mathbf{Y} \to \text{Cov}\left[\vec{X}, Y\right] \tag{2.18}$$

$$\frac{1}{n}\mathbf{X}^T\mathbf{X} \to \mathbf{v} \tag{2.19}$$

where as before $\mathbf{X}$ is the data-frame matrix with one row for each data point and one column for each variable, and similarly for $\mathbf{Y}$.

So, by continuity,

$$\widehat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} \to \beta \tag{2.20}$$

and we have a consistent estimator.

On the other hand, we could start with the empirical or in-sample mean squared error

$$\text{MSE}(\beta) \equiv \frac{1}{n}\sum_{i=1}^{n}(y_i - \vec{x}_i \cdot \beta)^2 \tag{2.21}$$

and minimize it. The minimizer is the same $\widehat{\beta}$ we got by plugging in the sample covariances. No probabilistic assumption is needed to minimize the in-sample MSE, but it doesn't let us say anything about the *convergence* of $\widehat{\beta}$. For that, we do need some assumptions about $\vec{X}$ and Y coming from distributions with unchanging covariances.

(One can also show that the least-squares estimate is the linear predictor with the minimax prediction risk. That is, its worst-case performance, when everything goes wrong and the data are horrible, will be better than any other linear method. This is some comfort, especially if you have a gloomy and pessimistic view of data, but other methods of estimation may work better in less-than-worst-case scenarios.)

2.1.3.1 Unbiasedness and Variance of Ordinary Least Squares Estimates

The very weak assumptions we have made still let us say a little bit more about the properties of the ordinary least squares estimate $\widehat{\beta}$. To do so, we need to think about why $\widehat{\beta}$ fluctuates. For the moment, let's fix $\mathbf{X}$ at a particular value $\mathbf{x}$, but allow $\mathbf{Y}$ to vary randomly (what's called "fixed design" regression). That means that $\beta = \mathbb{V}\left[\vec{X} \mid \mathbf{X} = \mathbf{x}\right]^{-1} \text{Cov}\left[\vec{X}, Y \mid \mathbf{X} = \mathbf{x}\right]$.

The key fact is that $\widehat{\beta}$ is linear in the observed responses $\mathbf{Y}$. We can use this by writing, as you're used to from your linear regression class,

$$Y = \vec{X} \cdot \beta + \epsilon \tag{2.22}$$

Here ϵ is the noise around the optimal linear predictor; we have to remember that while $\mathbb{E}[\epsilon] = 0$ and $\text{Cov}\left[\epsilon, \vec{X}\right] = 0$, it is not generally true that $\mathbb{E}\left[\epsilon \mid \vec{X} = \vec{x}\right] = 0$ or that $\mathbb{V}\left[\epsilon \mid \vec{X} = \vec{x}\right]$ is constant.

Let's assume for the moment that all the coordinates of $\vec{X}$ are centered. (Remember that centering doesn't change variances or covariances, which are what matter for β.) So $\mathbb{V}\left[\vec{X} \mid \mathbf{X} = \mathbf{x}\right] = n^{-1}\mathbf{x}^T\mathbf{x}$. Similarly, $\text{Cov}\left[\vec{X}, Y \mid \mathbf{X} = \mathbf{x}\right] = \mathbb{E}\left[\vec{X}Y \mid \mathbf{X} = \mathbf{x}\right] = n^{-1}\mathbf{x}^T\mathbb{E}\left[\mathbf{Y} \mid \mathbf{X} = \mathbf{x}\right]$. So

$$\begin{aligned}
\beta &= (n^{-1}\mathbf{x}^T\mathbf{x})^{-1}n^{-1}\mathbf{x}^T\mathbb{E}\left[\mathbf{Y} \mid \mathbf{X} = \mathbf{x}\right] && (2.23)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}\left[\mathbf{Y} \mid \mathbf{X} = \mathbf{x}\right] && (2.24)\\
&= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}\left[\mathbf{X}\beta + \epsilon \mid \mathbf{X} = \mathbf{x}\right] && (2.25)\\
&= \beta + (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}\left[\epsilon \mid \mathbf{X} = \mathbf{x}\right] && (2.26)\\
0 &= (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbb{E}\left[\epsilon \mid \mathbf{X} = \mathbf{x}\right] && (2.27)
\end{aligned}$$

Now we use the key fact that the estimate is linear in $\mathbf{Y}$:

$$\begin{aligned}
\widehat{\beta} &= \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\mathbf{Y} && (2.28)\\
&= \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T(\mathbf{x}\beta + \epsilon) && (2.29)\\
&= \beta + \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\epsilon && (2.30)
\end{aligned}$$

This directly tells us that $\widehat{\beta}$ is an unbiased estimate of β:

$$\begin{aligned}
\mathbb{E}\left[\widehat{\beta} \mid \mathbf{X} = \mathbf{x}\right] &= \beta + \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\mathbb{E}\left[\epsilon \mid \mathbf{X} = \mathbf{x}\right] && (2.31)\\
&= \beta + 0 = \beta && (2.32)
\end{aligned}$$

We can also get the variance matrix of $\widehat{\beta}$:

$$\begin{aligned}
\mathbb{V}\left[\widehat{\beta} \mid \mathbf{X} = \mathbf{x}\right] &= \mathbb{V}\left[\beta + \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\epsilon \mid \mathbf{x}\right] && (2.33)\\
&= \mathbb{V}\left[\left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\epsilon \mid \mathbf{X} = \mathbf{x}\right] && (2.34)\\
&= \left(\mathbf{x}^T\mathbf{x}\right)^{-1}\mathbf{x}^T\mathbb{V}\left[\epsilon \mid \mathbf{X} = \mathbf{x}\right]\mathbf{x}\left(\mathbf{x}^T\mathbf{x}\right)^{-1} && (2.35)
\end{aligned}$$

Let's write $\mathbb{V}\left[\epsilon \mid \mathbf{X} = \mathbf{x}\right]$ as a single matrix $\mathbf{\Sigma}(\mathbf{x})$. If the linear-prediction errors are uncorrelated with each other, then $\mathbf{\Sigma}$ will be diagonal. If they're also of equal variance, then $\mathbf{\Sigma} = \sigma^2\mathbf{I}$, and we have

$$\mathbb{V}\left[\widehat{\beta} \mid \mathbf{X} = \mathbf{x}\right] = \sigma^2(\mathbf{x}^T\mathbf{x})^{-1} = \frac{\sigma^2}{n}\left(\frac{1}{n}\mathbf{x}^T\mathbf{x}\right)^{-1} \quad (2.36)$$

Said in words, this means that the variance of our estimates of the linear-regression coefficient will (i) go down as the sample size n grows, (ii) go up as the linear regression gets worse (σ^2 grows), and (iii) go down as the regressors, the components of $\vec{X}$, have more sample variance themselves, and are less correlated with each other.

If we allow $\mathbf{X}$ to vary, then by the law of total variance,

$$\mathbb{V}\left[\widehat{\beta}\right] = \mathbb{E}\left[\mathbb{V}\left[\widehat{\beta} \mid X\right]\right] + \mathbb{V}\left[\mathbb{E}\left[\widehat{\beta} \mid X\right]\right] = \frac{\sigma^2}{n}\mathbb{E}\left[\left(\frac{1}{n}\mathbf{X}^T\mathbf{X}\right)^{-1}\right] \quad (2.37)$$

As $n \to \infty$, the sample variance matrix $n^{-1}\mathbf{X}^T\mathbf{X} \to \mathbf{v}$. Since matrix inversion is continuous, $\mathbb{V}\left[\widehat{\beta}\right] \to n^{-1}\sigma^2\mathbf{v}^{-1}$, and points (i)–(iii) still hold.

2.1.4 Some Geometry

When we have p regressors, we can think of our data as points in a $p+1$ dimensional space (regressors + Y). When we use a linear model, we are choosing to smooth or fit the data points on to a p-dimensional linear subspace, since that's what $\beta_0 + \vec{\beta} \cdot \vec{x}$ will give us. When we use least squares to chose the coefficients, we are minimizing the mean-square *vertical* distance between the subspace and the data points[7]. The linear subspace is a line when $p = 1$, a plane when $p = 2$, etc. I'll call this "the surface" for short.

The height of this surface, at a given value of $\vec{x}$, is our model's prediction for $\mathbb{E}\left[Y \mid \vec{X} = \vec{x}\right]$. If we're interested in a particular height, say y_0, we can set up the equation $y_0 = \beta_0 + \vec{\beta} \cdot \vec{x}$ and solve for $\vec{x}$. Since there's one equation, and p unknowns (the coordinates of $\vec{x}$), there isn't a unique solution; rather, the solutions themselves form a $(p-1)$-dimensional subspace. If we set up and solved the equation for a different value of y, say y_1, w'd get a different subspace, but it'd be parallel to the subspace we got for y_0. Every possible value of $\vec{x}$ is in one, and only one, of these parallel subspaces. If $p = 2$, the set of all $\vec{x}$ where $\beta_0 + \beta \cdot \vec{x} = y_0$ is just a line, and y_1 we get another line, parallel to the contour line for y_0, and every $\vec{x}$ is in one, and only one, contour line. Moving within a contour subspace (= contour line, when $p = 2$) doesn't change our prediction for Y, not matter how far we move. Moving from one contour surface to another does change our prediction for Y.

Remember, from vector algebra, that if we're looking at a dot or inner product between two vectors, say $\vec{c} \cdot \vec{d}$, we can break $\vec{d}$ up into two parts, $\vec{d} = \vec{d}_{\parallel} + \vec{d}_{\perp}$, where $\vec{d}_{\parallel}$ is parallel to $\vec{c}$ and $\vec{d}_{\perp}$ is perpendicular to $\vec{c}$, and then $\vec{c} \cdot \vec{d} = \vec{c} \cdot \vec{d}_{\parallel} = \|\vec{c}\|\|\vec{d}_{\parallel}\|$. Applied to the inner product in a linear model, $\vec{\beta} \cdot \vec{x}$, this tells us that only the part of $\vec{x}$ which is parallel to $\vec{\beta}$ matters for the prediction. $\vec{\beta}$ gives us the direction in the p-dimensional space of regressors which matters to the linear model. Moving $\vec{x}$ back and forth in this direction changes our prediction. Moving $\vec{x}$ in any of the $p-1$ other, orthogonal directions, no matter how far we move it, does absolutely nothing to the linear model's predictions.

The last paragraph repeated phrases like "matters to the linear model" quite tiresomely, but it did so for a reason. *If* the relationship between Y and $\vec{X}$ really is linear, then $\vec{\beta}$ really *is* the only direction in regressor space that matters, we really *can* move arbitrarily far perpendicular to $\vec{\beta}$ without changing the expected value of Y, etc., etc. If the real relationship is nonlinear, though, none of that is true of reality, but it's still true of the linear model.

[7] If we want to minimize the mean-square *distance* between the data points and a linear subspace, we need to use principal components analysis, as explained in Chapter 15.

```
x1 <- runif(100)
x2 <- rnorm(100, 0.5, 0.1)
x3 <- runif(100, 2, 3)
y1 <- sqrt(x1) + rnorm(length(x1), 0, 0.05)
y2 <- sqrt(x2) + rnorm(length(x2), 0, 0.05)
y3 <- sqrt(x3) + rnorm(length(x3), 0, 0.05)
plot(x1, y1, xlim = c(0, 3), ylim = c(0, 3), xlab = "X", ylab = "Y", col = "darkgreen",
    pch = 15)
rug(x1, side = 1, col = "darkgreen")
rug(y1, side = 2, col = "darkgreen")
points(x2, y2, pch = 16, col = "blue")
rug(x2, side = 1, col = "blue")
rug(y2, side = 2, col = "blue")
points(x3, y3, pch = 17, col = "red")
rug(x3, side = 1, col = "red")
rug(y3, side = 2, col = "red")
lm1 <- lm(y1 ~ x1)
lm2 <- lm(y2 ~ x2)
lm3 <- lm(y3 ~ x3)
abline(lm1, col = "darkgreen", lty = "dotted")
abline(lm2, col = "blue", lty = "dashed")
abline(lm3, col = "red", lty = "dotdash")
x.all <- c(x1, x2, x3)
y.all <- c(y1, y2, y3)
lm.all <- lm(y.all ~ x.all)
abline(lm.all, lty = "solid")
curve(sqrt(x), col = "grey", add = TRUE)
legend("topleft", legend = c("Unif[0,1]", "N(0.5, 0.01)", "Unif[2,3]", "Union of above",
    "True regression line"), col = c("black", "blue", "red", "black", "grey"), pch = c(15,
    16, 17, NA, NA), lty = c("dotted", "dashed", "dotdash", "solid", "solid"))
```

Code Example 1: *Code used to make Figure 2.1.*

2.2 Shifting Distributions, Omitted Variables, and Transformations

2.2.1 Changing Slopes

I said earlier that the best β in linear regression will depend on the distribution of the regressors, unless the conditional mean is exactly linear. Here is an illustration. For simplicity, let's say that $p = 1$, so there's only one regressor. I generated data from $Y = \sqrt{X} + \epsilon$, with $\epsilon \sim \mathcal{N}(0, 0.05^2)$ (i.e. the standard deviation of the noise was 0.05). Figure 2.1 shows the lines inferred from samples with three different distributions of X: $X \sim \text{Unif}(0, 1)$, $X \sim \mathcal{N}(0.5, 0.01)$, and $X \sim \text{Unif}(2, 3)$. Some distributions of X lead to similar (and similarly wrong) regression lines; doing one estimate from all three data sets gives yet another answer.

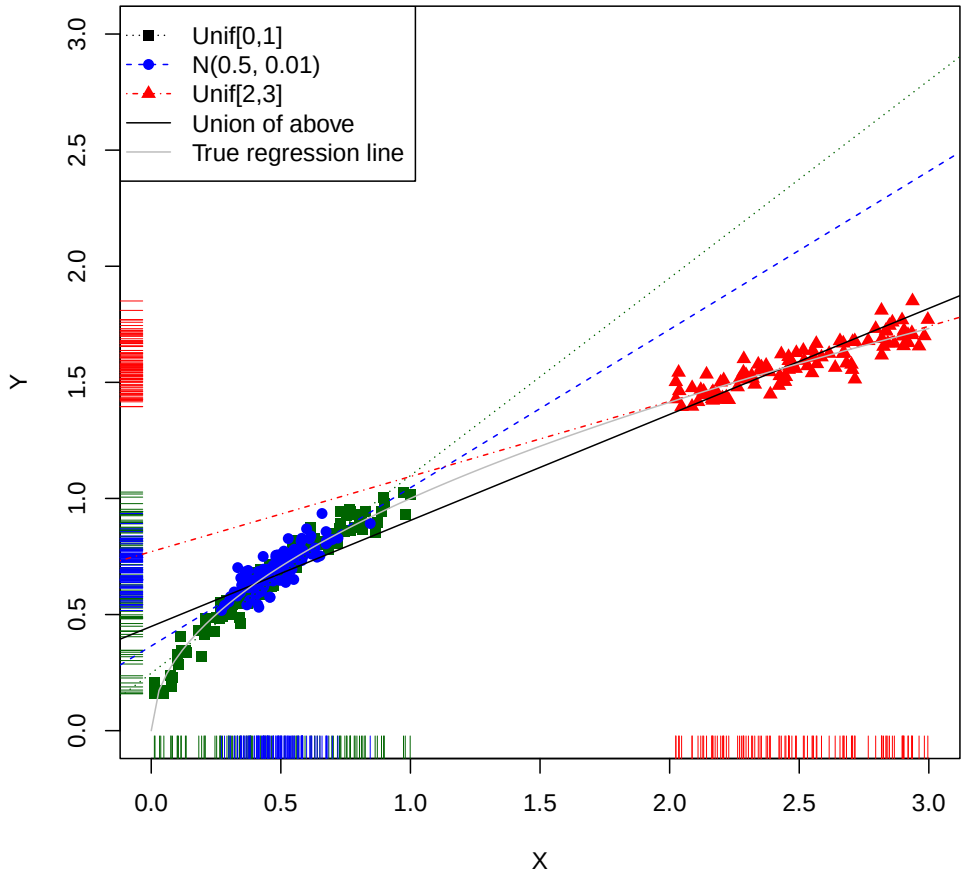


Figure 2.1 Behavior of the conditional distribution $Y \mid X \sim \mathcal{N}(\sqrt{X}, 0.05^2)$ with different distributions of X. The dots (in different colors and shapes) show three different distributions of X (with sample values indicated by colored "rug" ticks on the axes), plus the corresponding regression lines. The solid line is the regression using all three sets of points, and the grey curve is the true regression function. (See Code Example 1 for the code use to make this figure.) Notice how different distributions of X give rise to different slopes, each of which may make sense as a local approximation to the truth.

2.2.1.1 R^2: Distraction or Nuisance?

This little set-up, by the way, illustrates that R^2 is not a stable property of the distribution either. For the black points, $R^2 = 0.92$; for the blue, $R^2 = 0.70$; and for the red, $R^2 = 0.77$; and for the complete data, 0.96. Other sets of x_i values would give other values for R^2. Note that while the global linear fit isn't even a good approximation anywhere in particular, it has the highest R^2.

This kind of perversity can happen even in a completely linear set-up. Suppose now that $Y = aX + \epsilon$, and we happen to know a exactly. The variance of Y will be $a^2\mathbb{V}[X] + \mathbb{V}[\epsilon]$. The amount of variance our regression "explains" — really, the variance of our predictions — will be $a^2\mathbb{V}[X]$. So $R^2 = \frac{a^2\mathbb{V}[X]}{a^2\mathbb{V}[X]+\mathbb{V}[\epsilon]}$. This goes to zero as $\mathbb{V}[X] \to 0$ and it goes to 1 as $\mathbb{V}[X] \to \infty$. It thus has little to do with the quality of the fit, and a lot to do with how spread out the regressor is.

Notice also how easy it is to get a very high R^2 even when the true model is not linear!

2.2.2 Omitted Variables and Shifting Distributions

That the optimal regression coefficients can change with the distribution of the predictor features is annoying, but one could after all *notice* that the distribution has shifted, and so be cautious about relying on the old regression. More subtle is that the regression coefficients can depend on variables which you do not measure, and those can shift without your noticing anything.

Mathematically, the issue is that

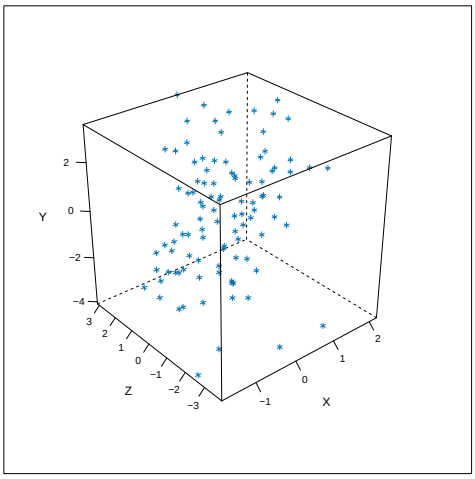
$$\mathbb{E}\left[Y \mid \vec{X}\right] = \mathbb{E}\left[\mathbb{E}\left[Y \mid Z, \vec{X}\right] \mid \vec{X}\right] \tag{2.38}$$

Now, if Y is independent of Z given $\vec{X}$, then the extra conditioning in the inner expectation does nothing and changing Z doesn't alter our predictions. But in general there will be plenty of variables Z which we don't measure (so they're not included in $\vec{X}$) but which have some non-redundant information about the response (so that Y depends on Z even conditional on $\vec{X}$). If the distribution of $\vec{X}$ given Z changes, then the optimal regression of Y on $\vec{X}$ should change too.

Here's an example. X and Z are both $\mathcal{N}(0,1)$, but with a positive correlation of 0.1. In reality, $Y \sim \mathcal{N}(X + Z, 0.01)$. Figure 2.2 shows a scatterplot of all three variables together ($n = 100$).

Now I change the correlation between X and Z to -0.1. This leaves both marginal distributions alone, and is barely detectable by eye (Figure 2.3).

Figure 2.4 shows just the X and Y values from the two data sets, in black for the points with a positive correlation between X and Z, and in blue when the correlation is negative. Looking by eye at the points and at the axis tick-marks, one sees that, as promised, there is very little change in the *marginal* distribution of either variable. Furthermore, the correlation between X and Y doesn't change much, going only from 0.7 to 0.59. On the other hand, the regression lines are noticeably different. When $\mathrm{Cov}[X, Z] = 0.1$, the slope of the regression line is 1.2



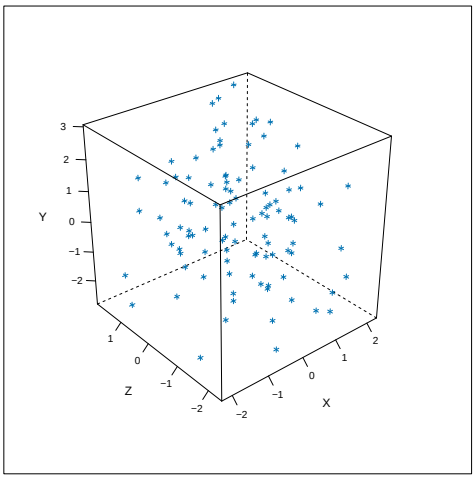

```
library(lattice)
library(MASS)

x.z = mvrnorm(100, c(0, 0), matrix(c(1, 0.1, 0.1, 1), nrow = 2))
y = x.z[, 1] + x.z[, 2] + rnorm(100, 0, 0.1)
cloud(y ~ x.z[, 1] * x.z[, 2], xlab = "X", ylab = "Z", zlab = "Y", scales = list(arrows = FALSE),
    col.point = "black")
```

Figure 2.2 Scatter-plot of response variable Y (vertical axis) and two variables which influence it (horizontal axes): X, which is included in the regression, and Z, which is omitted. X and Z have a correlation of $+0.1$.

— high values for X tend to indicate high values for Z, which also increases Y. When $\mathrm{Cov}\left[X, Z\right] = -0.1$, the slope of the regression line is 0.79, since extreme values of X are now signs that Z is at the opposite extreme, bringing Y closer back to its mean. But, to repeat, the difference is due to changing the correlation between X and Z, not how X and Z themselves relate to Y. If I regress Y on X and Z, I get $\widehat{\beta} = 1, 1$ in the first case and $\widehat{\beta} = 1, 0.99$ in the second.

We'll return to omitted variables when we look at causal inference in Part III.




```
new.x.z = mvrnorm(100, c(0, 0), matrix(c(1, -0.1, -0.1, 1), nrow = 2))
new.y = new.x.z[, 1] + new.x.z[, 2] + rnorm(100, 0, 0.1)
cloud(new.y ~ new.x.z[, 1] * new.x.z[, 2], xlab = "X", ylab = "Z", zlab = "Y", scales = list(arrows
```

Figure 2.3 As in Figure 2.2, but shifting so that the correlation between X and Z is now -0.1, though the marginal distributions, and the distribution of Y given X and Z, are unchanged.

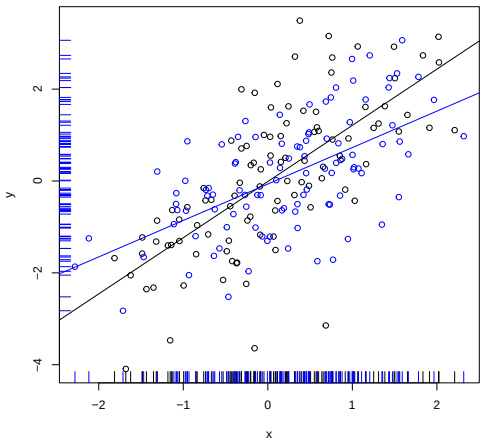


Figure 2.4 Joint distribution of X and Y from Figure 2.2 (black, with a positive correlation between X and Z) and from Figure 2.3 (blue, with a negative correlation between X and Z). Tick-marks on the axes show the marginal distributions, which are manifestly little-changed. (See accompanying R file for commands.)

2.2.3 Errors in Variables

Often, the predictor variables we can actually measure, $\vec{X}$, are distorted versions of some other variables $\vec{U}$ we wish we could measure, but can't:

$$\vec{X} = \vec{U} + \vec{\eta} \tag{2.39}$$

with $\vec{\eta}$ being some sort of noise. Regressing Y on $\vec{X}$ then gives us what's called an **errors-in-variables** problem.

In one sense, the errors-in-variables problem is huge. We are often much more interested in the connections between *actual* variables in the *real* world, than with our imperfect, noisy measurements of them. Endless ink has been spilled, for instance, on what determines students' test scores. One thing commonly thrown into the regression — a feature included in $\vec{X}$ — is the income of children's families. But this is rarely measured precisely[8], so what we are really interested in — the relationship between actual income and school performance — is not what our regression estimates. Typically, adding noise to the input features makes them less predictive of the response — in linear regression, it tends to push $\widehat{\beta}$ closer to zero than it would be if we could regress Y on $\vec{U}$.

On account of the error-in-variables problem, some people get very upset when they see imprecisely-measured features as inputs to a regression. Some of them, in fact, demand that the input variables be measured *exactly*, with no noise whatsoever. This position, however, is crazy, and indeed there's a sense in which errors-in-variables isn't a problem at all. Our earlier reasoning about how to find the optimal linear predictor of Y from $\vec{X}$ remains valid whether something like Eq. 2.39 is true or not. Similarly, the reasoning in Ch. 1 about the actual regression function being the over-all optimal predictor, etc., is unaffected. If we will continue to have $\vec{X}$ rather than $\vec{U}$ available to us for prediction, then Eq. 2.39 is irrelevant *for prediction*. Without better data, the relationship of Y to $\vec{U}$ is just one of the unanswerable questions the world is full of, as much as "what song the sirens sang, or what name Achilles took when he hid among the women".

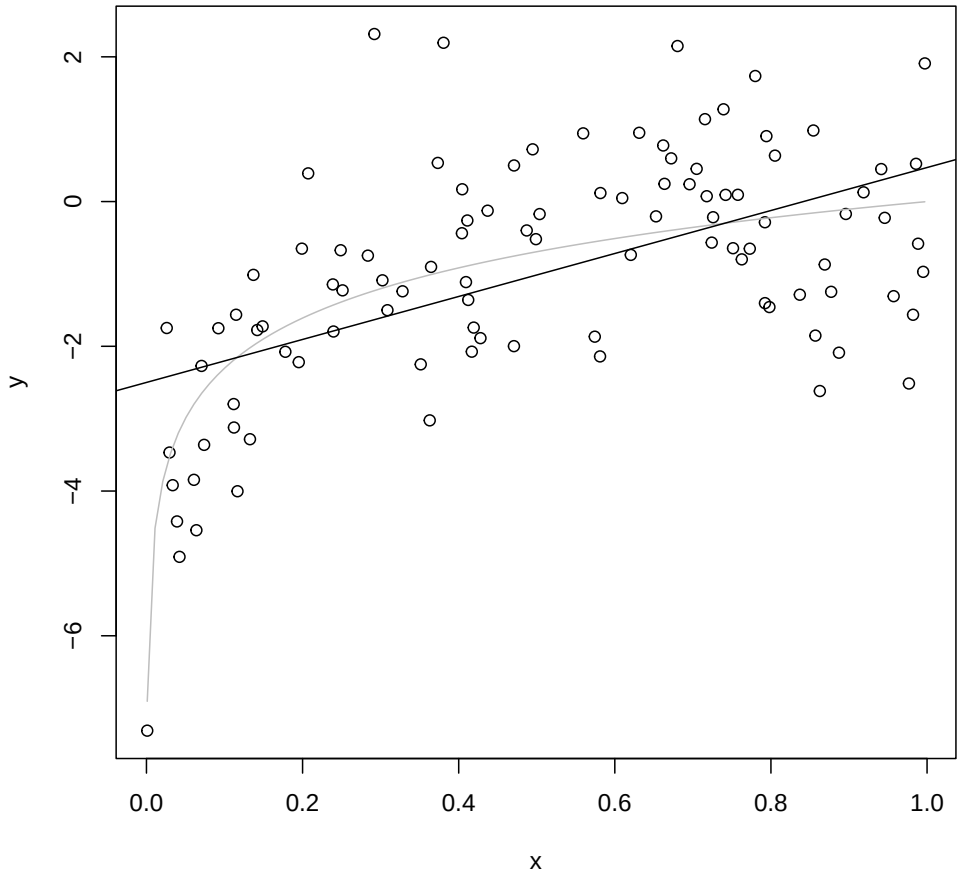
Now, if you are willing to assume that $\vec{\eta}$ is a *very* well-behaved Gaussian with known variance, then there are solutions to the error-in-variables problem for linear regression, i.e., ways of estimating the coefficients you'd get from regressing Y on $\vec{U}$. I'm not going to go over them, partly because they're in standard textbooks, but mostly because the assumptions are hopelessly demanding.[9]

2.2.4 Transformation

Let's look at a simple non-linear example, $Y \mid X \sim \mathcal{N}(\log X, 1)$. The problem with smoothing data like this on to a straight line is that the true regression curve isn't straight, $\mathbb{E}[Y \mid X = x] = \log x$. (Figure 2.5.) This suggests replacing

[8] One common proxy is to ask *the child* what they think their family income is. (I didn't believe that either when I first read about it.)

[9] Non-parametric error-in-variable methods are an active topic of research (Carroll *et al.*, 2009).




```
x <- runif(100)
y <- rnorm(100, mean = log(x), sd = 1)
plot(y ~ x)
curve(log(x), add = TRUE, col = "grey")
abline(lm(y ~ x))
```

Figure 2.5 Sample of data for $Y \mid X \sim \mathcal{N}(\log X, 1)$. (Here $X \sim \text{Unif}(0,1)$, and all logs are natural logs.) The true, logarithmic regression curve is shown in grey (because it's not really observable), and the linear regression fit is shown in black.

the variables we have with ones where the relationship *is* linear, and then undoing the transformation to get back to what we actually measure and care about.

We have two choices: we can transform the response Y, or the predictor X. Here

transforming the response would mean regressing $\exp Y$ on X, and transforming the predictor would mean regressing Y on $\log X$. Both kinds of transformations can be worth trying. The best reasons to use one kind rather than another are those that come from subject-matter knowledge: if we have good reason to think that that $f(Y) = \beta X + \epsilon$, then it can make a lot of sense to transform Y. If genuine subject-matter considerations are not available, however, my experience is that transforming the predictors, rather than the response, is a better bet, for several reasons.

1. Mathematically, $\mathbb{E}[f(Y)] \neq f(\mathbb{E}[Y])$. A mean-squared optimal prediction of $f(Y)$ is not necessarily close to the transformation of an optimal prediction of Y. And Y is, presumably, what we really want to predict.
2. Imagine that $Y = \sqrt{X} + \log Z$. There's not going to be any particularly nice transformation of Y that makes everything linear, though there will be transformations of the features. This generalizes to more complicated models with features built from multiple covariates.
3. Suppose that we are in luck and $Y = \mu(X) + \epsilon$, with ϵ independent of X, and Gaussian, so all the usual default calculations about statistical inference apply. Then it will generally *not* be the case that $f(Y) = s(X) + \eta$, with η a Gaussian random variable independent of X. In other words, transforming Y completely messes up the noise model. (Consider the simple case where we take the logarithm of Y. Gaussian noise after the transformation implies log-normal noise before the transformation. Conversely, Gaussian noise before the transformation implies a very weird, nameless noise distribution after the transformation.)

Figure 2.6 shows the effect of these transformations. Here transforming the predictor does, indeed, work out more nicely; but of course I chose the example so that it does so.

To expand on that last point, imagine a model like so:

$$\mu(\vec{x}) = \sum_{j=1}^{q} c_j f_j(\vec{x}) \tag{2.40}$$

If we know the functions f_j, we can estimate the optimal values of the coefficients c_j by least squares — this is a regression of the response on new features, which happen to be defined in terms of the old ones. Because the parameters are outside the functions, that part of the estimation works just like linear regression. Models embraced under the heading of Eq. 2.40 include linear regressions with **interactions** between the regressors (set $f_j = x_i x_k$, for various combinations of i and k), and **polynomial regression**. There is however nothing magical about using products and powers of the regressors; we could regress Y on $\sin x$, $\sin 2x$, $\sin 3x$, etc.

To apply models like Eq. 2.40, we can either (a) fix the functions f_j in advance, based on guesses about what should be good features for this problem; (b) fix the functions in advance by always using some "library" of mathematically convenient

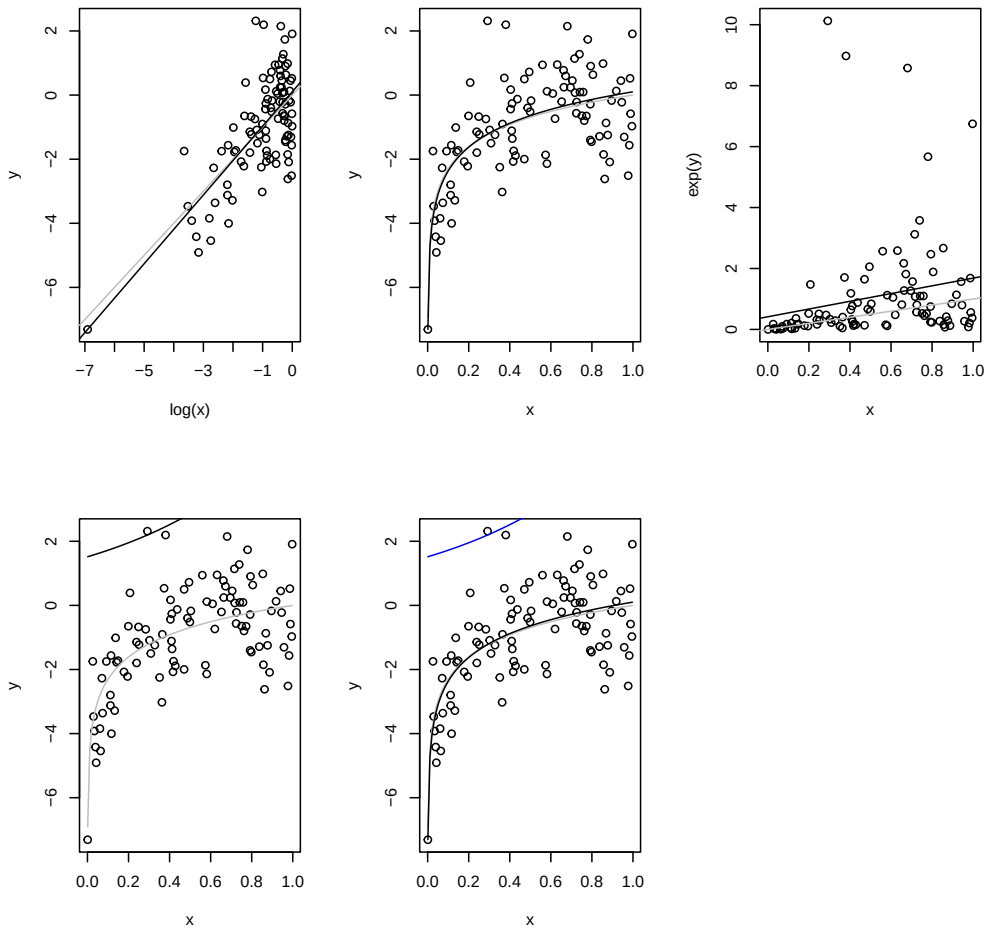


Figure 2.6 Transforming the predictor (left column) and the response (right) in the data from Figure 2.5, shown in both the transformed coordinates (top) and the original coordinates (middle). The bottom figure super-imposes the two estimated curves (transformed X in black, transformed Y in blue). The true regression curve is always in grey. (R code deliberately omitted; reproducing this is Exercise 2.4.)

functions, like polynomials or trigonometric functions; or (c) try to *find* good functions from the data. Option (c) takes us beyond the realm of linear regression as such, into things like **splines** (Chapter 7) and **additive models** (Chapter 8). It is also possible to search for transformations of *both* sides of a regression model; see Breiman and Friedman (1985) and, for an R implementation, Spector *et al.* (2013).

2.3 Adding Probabilistic Assumptions

The usual treatment of linear regression adds many more probabilistic assumptions, namely that

$$Y \mid \vec{X} \sim \mathcal{N}(\vec{X} \cdot \beta, \sigma^2) \tag{2.41}$$

and that Y values are independent conditional on their $\vec{X}$ values. So now we are *assuming* that the regression function is exactly linear; we are *assuming* that at each $\vec{X}$ the scatter of Y around the regression function is Gaussian; we are *assuming* that the variance of this scatter is constant; and we are *assuming* that there is no dependence between this scatter and anything else.

None of these assumptions was needed in deriving the optimal linear predictor. None of them is so mild that it should go without comment or without at least some attempt at testing.

Leaving that aside just for the moment, why make those assumptions? As you know from your earlier classes, they let us write down the likelihood of the observed responses $y_1, y_2, \ldots y_n$ (conditional on the covariates $\vec{x}_1, \ldots \vec{x}_n$), and then estimate β and σ^2 by maximizing this likelihood. As you also know, the maximum likelihood estimate of β is exactly the same as the β obtained by minimizing the residual sum of squares. This coincidence would not hold in other models, with non-Gaussian noise.

We saw earlier that $\widehat{\beta}$ is consistent under comparatively weak assumptions — that it converges to the optimal coefficients. But then there might, possibly, still be other estimators are also consistent, but which converge faster. If we make the extra statistical assumptions, so that $\widehat{\beta}$ is also the maximum likelihood estimate, we can lay that worry to rest. The MLE is generically (and certainly here!) **asymptotically efficient**, meaning that it converges as fast as any other consistent estimator, at least in the long run. So we are not, so to speak, wasting any of our data by using the MLE.

A further advantage of the MLE is that, as $n \to \infty$, its sampling distribution is itself a Gaussian, centered around the true parameter values. This lets us calculate standard errors and confidence intervals quite easily. Here, with the Gaussian assumptions, much more exact statements can be made about the distribution of $\widehat{\beta}$ around β. You can find the formulas in any textbook on regression, so I won't get into that.

We can also use a general property of MLEs for model testing. Suppose we have two classes of models, Ω and ω. Ω is the general case, with p parameters, and ω is a special case, where some of those parameters are constrained, but $q < p$ of them are left free to be estimated from the data. The constrained model class ω is then **nested** within Ω. Say that the MLEs with and without the constraints are, respectively, $\widehat{\Theta}$ and $\widehat{\theta}$, so the maximum log-likelihoods are $L(\widehat{\Theta})$ and $L(\widehat{\theta})$. Because it's a maximum over a larger parameter space, $L(\widehat{\Theta}) \geq L(\widehat{\theta})$. On the other hand, if the true model really is in ω, we'd expect the constrained and unconstrained estimates to be converging. It turns out that the difference in log-likelihoods has an asymptotic distribution which doesn't depend on any of the

model details, namely

$$2\left[L(\widehat{\Theta}) - L(\widehat{\theta})\right] \rightsquigarrow \chi^2_{p-q} \tag{2.42}$$

That is, a χ^2 distribution with one degree of freedom for each extra parameter in Ω (that's why they're called "degrees of freedom").[10]

This approach can be used to test particular restrictions on the model, and so it is sometimes used to assess whether certain variables influence the response. This, however, gets us into the concerns of the next section.

2.3.1 Examine the Residuals

By construction, the errors of the optimal linear predictor have expectation 0 and are uncorrelated with the regressors. Also by construction, the residuals of a *fitted* linear regression have sample mean 0, and are uncorrelated, in the sample, with the regressors.

If the usual probabilistic assumptions hold, however, the errors of the optimal linear predictor have many other properties as well.

1. The errors have a Gaussian distribution at each $\vec{x}$.
2. The errors have the *same* Gaussian distribution at each $\vec{x}$, i.e., they are *independent* of the regressors. In particular, they must have the same variance (i.e., they must be homoskedastic).
3. The errors are *independent of each other*. In particular, they must be *uncorrelated* with each other.

When these properties — Gaussianity, homoskedasticity, lack of correlation — hold, we say that the errors are **white noise**. They imply strongly related properties for the residuals: the residuals should be Gaussian, with variances and covariances given by the hat matrix, or more specifically by $\mathbf{I} - \mathbf{x}(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T$ (§1.5.3.2). This means that the residuals will not be *exactly* white noise, but they should be *close* to white noise. You should check this! If you find residuals which are a long way from being white noise, you should be extremely suspicious of your model. These tests are much more important than checking whether the coefficients are significantly different from zero.

Every time someone uses linear regression with the standard assumptions for inference and does *not* test whether the residuals are white noise, an angel loses its wings.

2.3.2 On Significant Coefficients

If all the usual distributional assumptions hold, then t-tests can be used to decide whether particular coefficients are statistically-significantly different from zero.

[10] If you assume the noise is Gaussian, the left-hand side of Eq. 2.42 can be written in terms of various residual sums of squares. However, the equation itself remains valid under other noise distributions, which just change the form of the likelihood function.

Pretty much any piece of statistical software, R very much included, reports the results of these tests automatically. It is far too common to seriously over-interpret those results, for a variety of reasons.

Begin with exactly what hypothesis is being tested when R (or whatever) runs those t-tests. Say, without loss of generality, that there are p predictor variables, $\vec{X} = (X_1, \ldots X_p)$, and that we are testing the coefficient on X_p. Then the null hypothesis is not just "$\beta_p = 0$", but "$\beta_p = 0$ in a linear, Gaussian-noise model which also includes $X_1, \ldots X_{p-1}$, and nothing else". The alternative hypothesis is not just "$\beta_p \neq 0$", but "$\beta_p \neq 0$ in a linear, Gaussian-noise model which also includes $X_1, \ldots X_{p-1}$, but nothing else". The optimal linear coefficient on X_p will depend not just on the relationship between X_p and the response Y, but also on which other variables are included in the model. The test checks whether adding X_p really improves predictions more than would be expected, under all these assumptions, if one is already using all the other variables, and only those other variables. It does not, cannot, test whether X_p is important in any absolute sense.

Even if you are willing to say "Yes, all I really want to know about this variable is whether adding it to the model really helps me predict in a linear approximation", remember that the question which a t-test answers is whether adding that variable will help *at all*. Of course, as you know from your regression class, and as we'll see in more detail in Chapter 3, expanding the model never hurts its performance on the *training* data. The point of the t-test is to gauge whether the improvement in prediction is small enough to be due to chance, or so large, *compared to what noise could produce*, that one could confidently say the variable adds *some* predictive ability. This has several implications which are insufficiently appreciated among users.

In the first place, tests on individual coefficients can seem to contradict tests on groups of coefficients. Adding multiple variables to the model could significantly improve the fit (as checked by, say, a partial F test), even if *none* of the coefficients is significant on its own. In fact, every single coefficient in the model could be insignificant, while the model as a whole is highly significant (i.e., better than a flat line).

In the second place, it's worth thinking about which variables will show up as statistically significant. Remember that the t-statistic is $\widehat{\beta}_i/\mathrm{se}(\widehat{\beta}_i)$, the ratio of the estimated coefficient to its standard error. We saw above that $\mathbb{V}\left[\widehat{\beta} \mid \mathbf{X} = \mathbf{x}\right] = \frac{\sigma^2}{n}(n^{-1}\mathbf{x}^T\mathbf{x})^{-1} \to n^{-1}\sigma^2\mathbf{v}^{-1}$. This means that the standard errors will shrink as the sample size grows, so more and more variables will become significant as we get more data — but how much data we collect is irrelevant to how the process we're studying actually works. Moreover, at a fixed sample size, the coefficients with smaller standard errors will tend to be the ones whose variables have more variance, and whose variables are less correlated with the other predictors. High input variance and low correlation help us *estimate* the coefficient precisely, but, again, they have nothing to do with whether the input variable actually *influences* the response a lot.

To sum up, it is *never* the case that statistical significance is the same as

scientific, real-world significance. The most important variables are *not* those with the largest-magnitude t statistics or smallest p-values. Statistical significance is always about what "signals" can be picked out clearly from background noise[11]. In the case of linear regression coefficients, statistical significance runs together the size of the coefficients, how bad the linear regression model is, the sample size, the variance in the input variable, and the correlation of that variable with all the others.

Of course, even the limited "does it help linear predictions enough to bother with?" utility of the usual t-test (and F-test) calculations goes away if the standard distributional assumptions do not hold, so that the calculated p-values are just wrong. One can sometimes get away with using bootstrapping (Chapter 6) to get accurate p-values for standard tests under non-standard conditions.

2.4 Linear Regression Is Not the Philosopher's Stone

The philosopher's stone, remember, was supposed to be able to transmute base metals (e.g., lead) into the perfect metal, gold (Eliade, 1971). Many people treat linear regression as though it had a similar ability to transmute a correlation matrix into a scientific theory. In particular, people often argue that:

1. because a variable has a significant regression coefficient, it must influence the response;
2. because a variable has an insignificant regression coefficient, it must not influence the response;
3. if the input variables change, we can predict how much the response will change by plugging in to the regression.

All of this is wrong, or at best right only under very particular circumstances.

We have already seen examples where influential variables have regression coefficients of zero. We have also seen examples of situations where a variable with no influence has a non-zero coefficient (e.g., because it is correlated with an omitted variable which does have influence). *If* there are no nonlinearities and *if* there are no omitted influential variables and *if* the noise terms are always independent of the predictor variables, are we good?

No. Remember from Equation 2.6 that the optimal regression coefficients depend on both the marginal distribution of the predictors and the joint distribution (covariances) of the response and the predictors. There is no reason whatsoever to suppose that if we *change* the system, this will leave the conditional distribution of the response alone.

A simple example may drive the point home. Suppose we surveyed all the cars in Pittsburgh, recording the maximum speed they reach over a week, and how often they are waxed and polished. I don't think anyone doubts that there will be a positive correlation here, and in fact that there will be a positive regression

[11] In retrospect, it might have been clearer to say "statistically *detectable*" rather than "statistically *significant*".

coefficient, even if we add in many other variables as predictors. Let us even postulate that the relationship is linear (perhaps after a suitable transformation). Would anyone believe that polishing cars will make them go faster? Manifestly not. But this is exactly how people interpret regressions in all kinds of applied fields — instead of saying polishing makes cars go faster, it might be saying that receiving targeted ads makes customers buy more, or that consuming dairy foods makes diabetes progress faster, or .... Those claims might be *true*, but the regressions could easily come out the same way were the claims false. Hence, the regression results provide little or no *evidence* for the claims.

Similar remarks apply to the idea of using regression to "control for" extra variables. If we are interested in the relationship between one predictor, or a few predictors, and the response, it is common to add a bunch of other variables to the regression, to check both whether the apparent relationship might be due to correlations with something else, and to "control for" those other variables. The regression coefficient is interpreted as how much the response would change, on average, if the predictor variable were increased by one unit, "holding everything else constant". There is a very particular sense in which this is true: it's a prediction about the difference in expected responses (conditional on the given values for the other predictors), assuming that the form of the regression model is right, *and* that observations are randomly drawn from the same population we used to fit the regression.

In a word, what regression does is *probabilistic* prediction. It says what will happen if we keep drawing from the same population, but *select* a sub-set of the observations, namely those with given values of the regressors. A **causal** or **counter-factual** prediction would say what would happen if we (or Someone) *made* those variables take those values. Sometimes there's no difference between selection and intervention, in which case regression works as a tool for causal inference[12]; but in general there is. Probabilistic prediction is a worthwhile endeavor, but it's important to be clear that this is what regression does. There are techniques for doing causal prediction, which we will explore in Part III.

Every time someone thoughtlessly uses regression for causal inference, an angel not only loses its wings, but is cast out of Heaven and falls in extremest agony into the everlasting fire.

[12] In particular, if our model was estimated from data where Someone *assigned* values of the predictor variables in a way which breaks possible dependencies with omitted variables and noise — either by randomization or by experimental control — then regression can, in fact, work for causal inference.

2.5 Further Reading

If you would like to read a lot more — about 400 pages more — about linear regression from this perspective, see *The Truth About Linear Regression*, at `http://www.stat.cmu.edu/~cshalizi/TALR/`. That manuscript began as class notes for the class *before* this one, and has some overlap.

There are many excellent textbooks on linear regression. Among them, I would mention Weisberg (1985) for general statistical good sense, along with Faraway (2004) for R practicalities, and Hastie *et al.* (2009) for emphasizing connections to more advanced methods. Berk (2004) omits the details those books cover, but is superb on the big picture, and especially on what must be assumed in order to do certain things with linear regression and what cannot be done under any assumption.

For some of the story of how the usual probabilistic assumptions came to have that status, see, e.g., Lehmann (2008). On the severe issues which arise for the usual inferential formulas when the model is incorrect, see Buja *et al.* (2014).

Linear regression is a special case of both additive models (Chapter 8), and of locally linear models (§10.5). In most practical situations, additive models are a better idea than linear ones.

Historical notes

Because linear regression is such a big part of statistical practice, its history has been extensively treated in general histories of statistics, such as Stigler (1986) and Porter (1986). Farebrother (1999) is especially clear on transition from the first appearance of the method of least squares, where it was used to find parameters when there were more equations than unknowns[13], to more general linear modeling. I would particularly recommend Klein (1997) for a careful account of how regression, on its face a method for doing comparisons at one time across a population, came to be used to study causality and dynamics. The paper by Lehmann (2008) mentioned earlier is also informative.

The derivation of the optimal linear predictor in §2.1, assuming nothing beyond wanting to use a linear prediction function and $\mathbf{v}$ being invertible, is standard in

[13] The classic cases where astronomy and "geodesy", the measurement of the exact shape of the Earth (important for physics and for navigation). Take astronomy: if you have a model of the orbit of a planet, and plug in values for the parameters, you get a prediction for the the (apparent) position of the planet in the sky every night. Going the other direction, every observation gives you an equation with the unknown parameters on one side, and known, measured values on the other side. Even with a very complicated model with dozens of adjustable parameters, a few years worth of nightly observations gives you more equations than unknowns. With more equations than unknowns, there's usually no solution that fits all the data exactly. The literally-ancient approach to this embarrassing problem, going back to the ancient Greeks and Babylonians, was to try to select the *best*, most reliable observations, discarding the bad ones until you had just as many observations as unknowns, and then solving for the parameters. The crucial innovation in the 1700s was to realize that least squares gave us a way of trying to use *all* the observations, giving parameter values that *generally* fit well but not perfectly, because even the best observations are imperfect. In this context, the emphasis on *linear* equations made sense, because of the form of the models the astronomers and geodesists were using.

the theory of time series (Ch, 23) and stochastic processes, going back there at least to Kolmogorov (1941) and Wiener (1949). Special cases were known in the 1930s in factor analysis (Ch. 16), though I believe all of them also, unnecessarily, assumed Gaussian distributions for all variables. It's possible someone else got there first, but if so, I haven't been able to find it. In spatial statistics, the same ideas were re-discovered by D. G. Krige in the 1950s (Krige, 1981), and popularized by Georges Matheron under the name "kriging" (Matheron, 2019), which has stuck in geostatistics.

Exercises

2.1 1. Write the expected squared error of a linear predictor with slopes $\vec{b}$ and intercept b_0 as a function of those coefficients.
    2. Find the derivatives of the expected squared error with respect to all the coefficients.
    3. Show that when we set all the derivatives to zero, the solutions are Eq. 2.6 and 2.5.

2.2 Show that the expected error of the optimal linear predictor, $\mathbb{E}\left[Y - \vec{X} \cdot \beta\right]$, is zero.

2.3 Convince yourself that if the real regression function is linear, β does not depend on the marginal distribution of X. You may want to start with the case of one predictor variable.

2.4 Run the code from Figure 2.5. Then replicate the plots in Figure 2.6.

2.5 Which kind of transformation is superior for the model where $Y \mid X \sim \mathcal{N}(\sqrt{X}, 1)$?

3

Evaluating Statistical Models: Error and Inference

3.1 What Are Statistical Models For? Summaries, Forecasts, Simulators

There are (at least) three ways we can use statistical models in data analysis: as summaries of the data, as predictors, and as simulators.

The least demanding use of a model is to summarize the data — to use it for **data reduction**, or **compression**. Just as the sample mean or sample quantiles can be descriptive statistics, recording some features of the data and saying nothing about a population or a generative process, we could use estimates of a model's parameters as descriptive summaries. Rather than remembering all the points on a scatter-plot, say, we'd just remember what the OLS regression surface was.

It's hard to be wrong about a summary, unless we just make a mistake. (It may not be *helpful* for us later, but that's different.) When we say "the slope which minimized the sum of squares was 4.02", we make no claims about anything *but* the training data. That statement relies on no assumptions, beyond our calculating correctly. But it also asserts nothing about the rest of the world. As soon as we try to connect our training data to anything else, we start relying on assumptions, and we run the risk of being wrong.

Probably the most common connection to want to make is to say what *other* data will look like — to make predictions. In a statistical model, with random variables, we do not anticipate that our predictions will ever be *exactly* right, but we also anticipate that our mistakes will show stable probabilistic patterns. We can evaluate predictions based on those patterns of error — how big is our typical mistake? are we biased in a particular direction? do we make a lot of little errors or a few huge ones?

Statistical inference about model parameters — estimation and hypothesis testing — can be seen as a kind of prediction, extrapolating from what we saw in a small piece of data to what we would see in the whole population, or whole process. When we *estimate* the regression coefficient $\hat{b} = 4.02$, that involves predicting new values of the dependent variable, but also predicting that if we repeated the experiment and re-estimated $\hat{b}$, we'd get a value close to 4.02.

Using a model to summarize old data, or to predict new data, doesn't commit us to assuming that the model describes the process which generates the data. But we often want to do that, because we want to interpret parts of the model

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

as aspects of the real world. We think that in neighborhoods where people have more money, they spend more on houses — perhaps each extra $1000 in income translates into an extra $4020 in house prices. Used this way, statistical models become stories about how the data were generated. If they are accurate, we should be able to use them to *simulate* that process, to step through it and produce something that looks, probabilistically, just like the actual data. This is often what people have in mind when they talk about *scientific* models, rather than just statistical ones.

An example: if you want to predict where in the night sky the planets will be, you can actually do very well with a model where the Earth is at the center of the universe, and the Sun and everything else revolve around it. You can even estimate, from data, how fast Mars (for example) goes around the Earth, or where, in this model, it should be tonight. But, since the Earth is *not* at the center of the solar system, those parameters don't actually refer to anything in reality. They are just mathematical fictions. On the other hand, we can also predict where the planets will appear in the sky using models where all the planets orbit the Sun, and the parameters of the orbit of Mars in that model *do* refer to reality.[1]

This chapter focuses on evaluating predictions, for three reasons. First, often we just want prediction. Second, if a model can't even predict well, it's hard to see how it could be right scientifically. Third, often the best way of checking a scientific model is to turn some of its implications into statistical predictions.

3.2 Errors, In and Out of Sample

With any predictive model, we can gauge how well it works by looking at its errors. We want these to be small; if they can't be small all the time we'd like them to be small on average. We may also want them to be patternless or unsystematic (because if there was a pattern to them, why not adjust for that, and make smaller mistakes). We'll come back to patterns in errors later, when we look at specification testing (Chapter 9). For now, we'll concentrate on the size of the errors.

To be a little more mathematical, we have a data set with points $\mathbf{z}_n = z_1, z_2, \ldots z_n$. (For regression problems, think of each data point as the pair of input and output values, so $z_i = (x_i, y_i)$, with x_i possibly a vector.) We also have various possible models, each with different parameter settings, conventionally written θ. For regression, θ tells us which regression function to use, so $m_\theta(x)$ or $m(x;\theta)$ is the prediction we make at point x with parameters set to θ. Finally, we have a **loss function** L which tells us how big the error is when we use a certain θ on a certain data point, $L(z, \theta)$. For mean-squared error, this would just be

$$L(z, \theta) = (y - m_\theta(x))^2 \tag{3.1}$$

[1] We can be pretty sure of this, because we use our parameter estimates to send our robots to Mars, and they get there.

But we could also use the mean absolute error

$$L(z, \theta) = |y - m_\theta(x)| \tag{3.2}$$

or many other loss functions. Sometimes we will actually be able to measure how costly our mistakes are, in dollars or harm to patients. If we had a model which gave us a distribution for the data, then $p_\theta(z)$ would a probability density at z, and a typical loss function would be the negative log-likelihood, $-\log m_\theta(z)$. No matter what the loss function is, I'll abbreviate the sample average of the loss over the whole data set by $\overline{L}(\mathbf{z}_n, \theta)$.

What we would like, ideally, is a predictive model which has zero error on future data. We basically never achieve this:

- The world just really is a noisy and stochastic place, and this means even the true, ideal model has non-zero error.[2] This corresponds to the first, σ_x^2, term in the bias-variance decomposition, Eq. 1.28 from Chapter 1.
- Our models are usually more or less **mis-specified**, or, in plain words, wrong. We hardly ever get the functional form of the regression, the distribution of the noise, the form of the causal dependence between two factors, etc., *exactly* right.[3] This is the origin of the bias term in the bias-variance decomposition. Of course we can get any of the details in the model specification *more or less* wrong, and we'd prefer to be less wrong.
- Our models are never perfectly estimated. Even if our data come from a perfect IID source, we only ever have a finite sample, and so our parameter estimates are (almost!) never quite the true, infinite-limit values. This is the origin of the variance term in the bias-variance decomposition. But as we get more and more data, the sample should become more and more representative of the whole process, and estimates should converge too.

So, because our models are flawed, we have limited data and the world is stochastic, we cannot expect even the best model to have zero error. Instead, we would like to minimize the **expected error**, or **risk**, or **generalization error**, on new data.

What we would like to do is to minimize the risk or expected loss

$$\mathbb{E}\left[L(Z, \theta)\right] = \int L(z, \theta) p(z) dz \tag{3.3}$$

To do this, however, we'd have to be able to calculate that expectation. Doing that would mean knowing the distribution of Z — the joint distribution of X and Y, for the regression problem. Since we don't know the true joint distribution, we need to approximate it somehow.

A natural approximation is to use our training data $\mathbf{z}_n$. For each possible model

[2] This is so even if you believe in some kind of ultimate determinism, because the variables we plug in to our predictive models are not complete descriptions of the physical state of the universe, but rather immensely coarser, and this coarseness shows up as randomness.

[3] Except maybe in fundamental physics, and even there our predictions are about our fundamental theories *in the context of experimental set-ups*, which we never model in complete detail.

θ, we can could calculate the sample mean of the error on the data, $\overline{L}(\mathbf{z}_n, \theta)$, called the **in-sample loss** or the **empirical risk**. The simplest strategy for estimation is then to pick the model, the value of θ, which minimizes the in-sample loss. This strategy is imaginatively called **empirical risk minimization**. Formally,

$$\widehat{\theta_n} \equiv \operatorname*{argmin}_{\theta \in \Theta} \overline{L}(\mathbf{z}_n, \theta) \tag{3.4}$$

This means picking the regression which minimizes the sum of squared errors, or the density with the highest likelihood[4]. This is what you've usually done in statistics courses so far, and it's very natural, but it does have some issues, notably optimism and over-fitting.

The problem of optimism comes from the fact that our training data isn't perfectly representative. The in-sample loss is a sample average. By the law of large numbers, then, we anticipate that, for each θ,

$$\overline{L}(\mathbf{z}_n, \theta) \to \mathbb{E}\left[L(Z, \theta)\right] \tag{3.5}$$

as $n \to \infty$. This means that, with enough data, the in-sample error is a good approximation to the generalization error of any given model θ. (Big samples are representative of the underlying population or process.) But this does *not* mean that the in-sample performance of $\hat{\theta}$ tells us how well it will generalize, because we purposely picked it to match the training data $\mathbf{z}_n$. To see this, notice that the in-sample loss equals the risk plus sampling noise:

$$\overline{L}(\mathbf{z}_n, \theta) = \mathbb{E}\left[L(\mathbf{Z}, \theta)\right] + \eta_n(\theta) \tag{3.6}$$

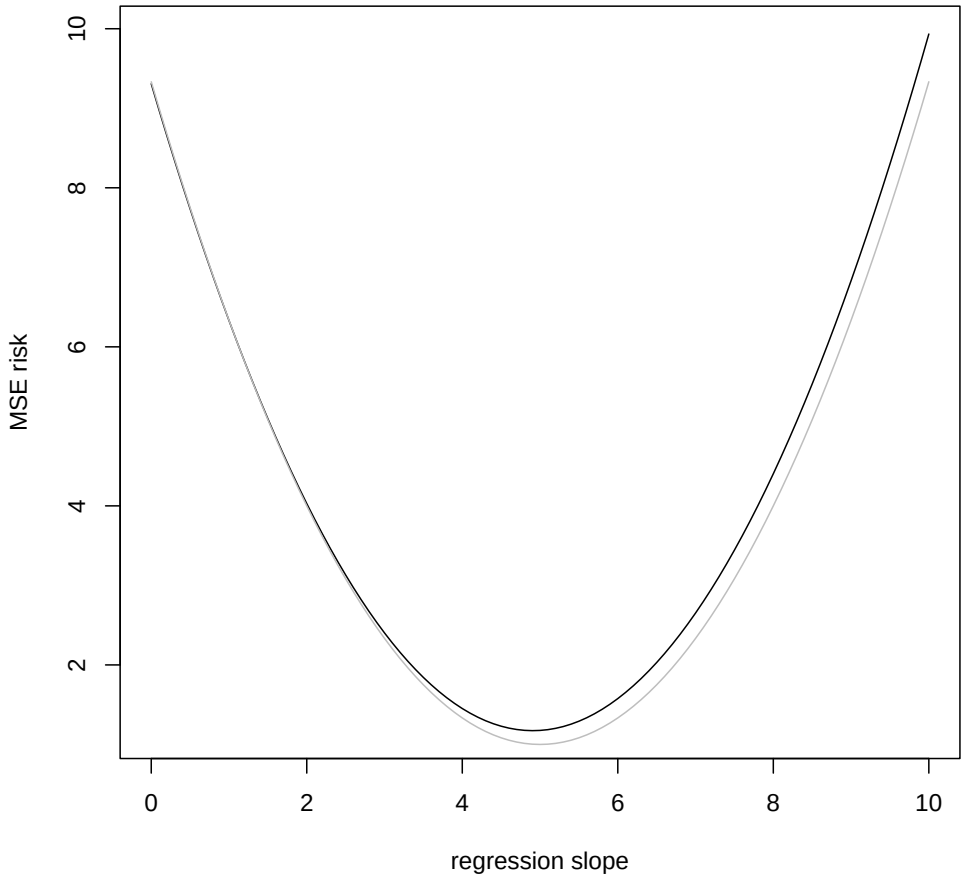
Here $\eta_n(\theta)$ is a random term which has mean zero, and represents the effects of having only a finite quantity of data, of size n, rather than the complete probability distribution. (I write it $\eta_n(\theta)$ as a reminder that different values of θ are going to be affected differently by the same sampling fluctuations.) The problem, then, is that the model which minimizes the in-sample loss could be one with good generalization performance ($\mathbb{E}\left[L(\mathbf{Z}, \theta)\right]$ is small), or it could be one which got very lucky ($\eta_n(\theta)$ was large and negative):

$$\widehat{\theta_n} = \operatorname*{argmin}_{\theta \in \Theta} \left(\mathbb{E}\left[L(Z, \theta)\right] + \eta_n(\theta)\right) \tag{3.7}$$

We only want to minimize $\mathbb{E}\left[L(Z, \theta)\right]$, but we can't separate it from $\eta_n(\theta)$, so we're almost surely going to end up picking a $\widehat{\theta_n}$ which was more or less lucky ($\eta_n < 0$) as well as good ($\mathbb{E}\left[L(Z, \theta)\right]$ small). This is the reason why picking the model which best fits the data tends to exaggerate how well it will do in the future (Figure 3.1).

Again, by the law of large numbers $\eta_n(\theta) \to 0$ for each θ, but now we need to worry about how fast it's going to zero, and whether that rate depends on θ. Suppose we knew that $\min_\theta \eta_n(\theta) \to 0$, or $\max_\theta |\eta_n(\theta)| \to 0$. Then it would

[4] Remember, maximizing the likelihood is the same as maximizing the log-likelihood, because log is an increasing function. Therefore maximizing the likelihood is the same as *minimizing* the *negative* log-likelihood.




```
n <- 20
theta <- 5
x <- runif(n)
y <- x * theta + rnorm(n)
empirical.risk <- function(b) {
    mean((y - b * x)^2)
}
true.risk <- function(b) {
    1 + (theta - b)^2 * (0.5^2 + 1/12)
}
curve(Vectorize(empirical.risk)(x), from = 0, to = 2 * theta, xlab = "regression slope",
    ylab = "MSE risk")
curve(true.risk, add = TRUE, col = "grey")
```

Figure 3.1 Empirical and generalization risk for regression through the origin, $Y = \theta X + \epsilon$, $\epsilon \sim \mathcal{N}(0,1)$, with true $\theta = 5$, and $X \sim \text{Unif}(0,1)$. Black: MSE on a particular sample ($n = 20$) as a function of slope, minimized at $\hat{\theta} = 4.91$. Grey: true or generalization risk (Exercise 3.2). The gap between the curves is the text's $\eta_n(\theta)$.

follow that $\eta_n(\widehat{\theta_n}) \to 0$, and the over-optimism in using the in-sample error to approximate the generalization error would at least be shrinking. If we knew how fast $\max_\theta |\eta_n(\theta)|$ was going to zero, we could even say something about how much bigger the true risk was likely to be. A lot of more advanced statistics and machine learning theory is thus about uniform laws of large numbers (showing $\max_\theta |\eta_n(\theta)| \to 0$) and rates of convergence.

Learning theory is a beautiful, deep, and practically important subject, but also a subtle and involved one. (See §3.6 for references.) To stick closer to analyzing real data, and to not turn this into an advanced probability class, I will only talk about some more-or-less heuristic methods, which are good enough for many purposes.

3.3 Over-Fitting and Model Selection

The big problem with using the in-sample error is related to optimism, but at once trickier to grasp and more important. This is the problem of **over-fitting**. To illustrate it, let's start with Figure 3.2. This has the twenty X values from a Gaussian distribution, and $Y = 7X^2 - 0.5X + \epsilon$, $\epsilon \sim \mathcal{N}(0,1)$. That is, the true regression curve is a parabola, with additive and independent Gaussian noise. Let's try fitting this — but pretend that we didn't know that the curve was a parabola. We'll try fitting polynomials of different degrees in x — degree 0 (a flat line), degree 1 (a linear regression), degree 2 (quadratic regression), up through degree 9. Figure 3.3 shows the data with the polynomial curves, and Figure 3.4 shows the in-sample mean squared error as a function of the degree of the polynomial.

Notice that the in-sample error goes down as the degree of the polynomial increases; it has to. Every polynomial of degree p can also be written as a polynomial of degree $p+1$ (with a zero coefficient for x^{p+1}), so going to a higher-degree model can only reduce the in-sample error. Quite generally, in fact, as one uses more and more complex and flexible models, the in-sample error will get smaller and smaller.[5]

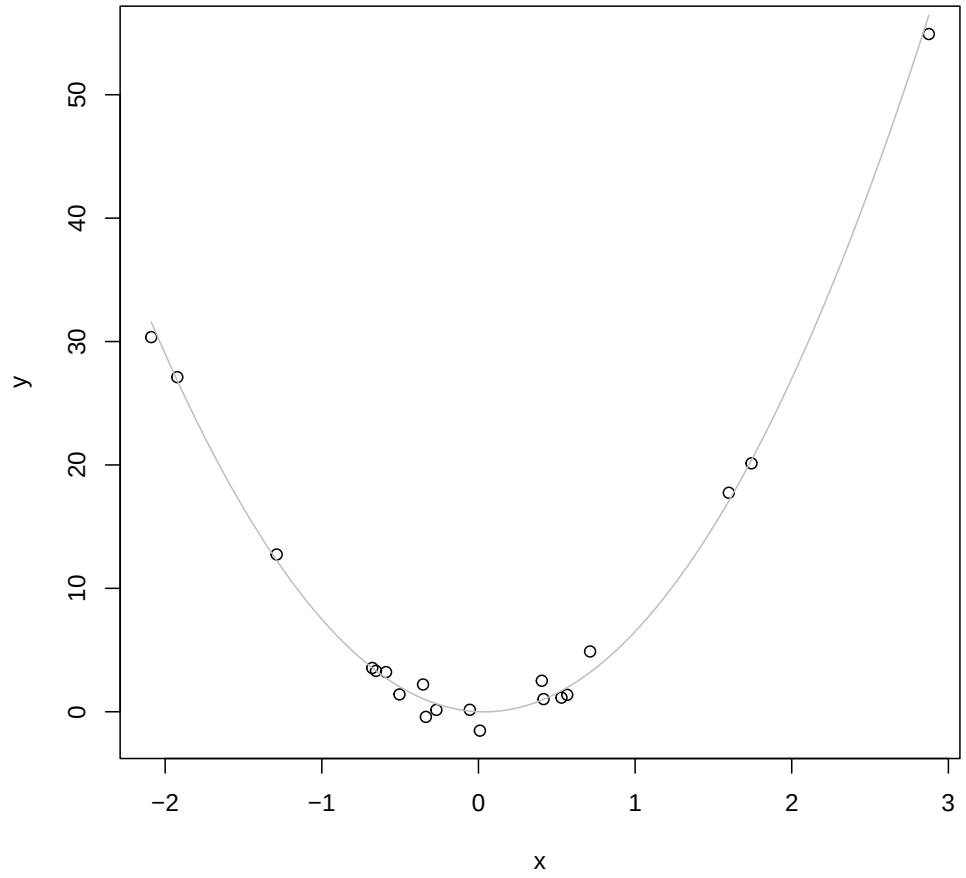
Things are quite different if we turn to the generalization error. In principle, I could calculate that for any of the models, since I know the true distribution, but it would involve calculating things like $\mathbb{E}[X^{18}]$, which won't be very illuminating. Instead, I will just draw a lot more data from the same source, twenty thousand data points in fact, and use the error of the old models on the new data as their generalization error[6]. The results are in Figure 3.5.

What is happening here is that the higher-degree polynomials — beyond degree 2 — are not just a *little* optimistic about how well they fit, they are *wildly* over-optimistic. The models which seemed to do notably better than a quadratic actually do much, much worse. If we picked a polynomial regression model based on in-sample fit, we'd chose the highest-degree polynomial available, and suffer for it.

In this example, the more complicated models — the higher-degree polynomials, with more terms and parameters — were not actually fitting the *generalizable* features of the data. Instead, they were fitting the sampling noise, the accidents which don't repeat. That is, the more complicated models **over-fit** the data. In terms of our earlier notation, η is bigger for the more flexible models. The model which does best here is the quadratic, because the true regression function happens to be of that form. The more powerful, more flexible, higher-degree polynomials were able to get closer to the training data, but that just meant

[5] In fact, since there are only 20 data points, they could all be fit exactly if the degree of the polynomials went up to 19. (Remember that any two points define a line, any three points a parabola, etc. — $p+1$ points define a polynomial of degree p which passes through them.)

[6] This works, yet again, because of the law of large numbers. In Chapters 5 and especially 6, we will see much more about replacing complicated probabilistic calculations with simple simulations, an idea sometimes called the "Monte Carlo method".



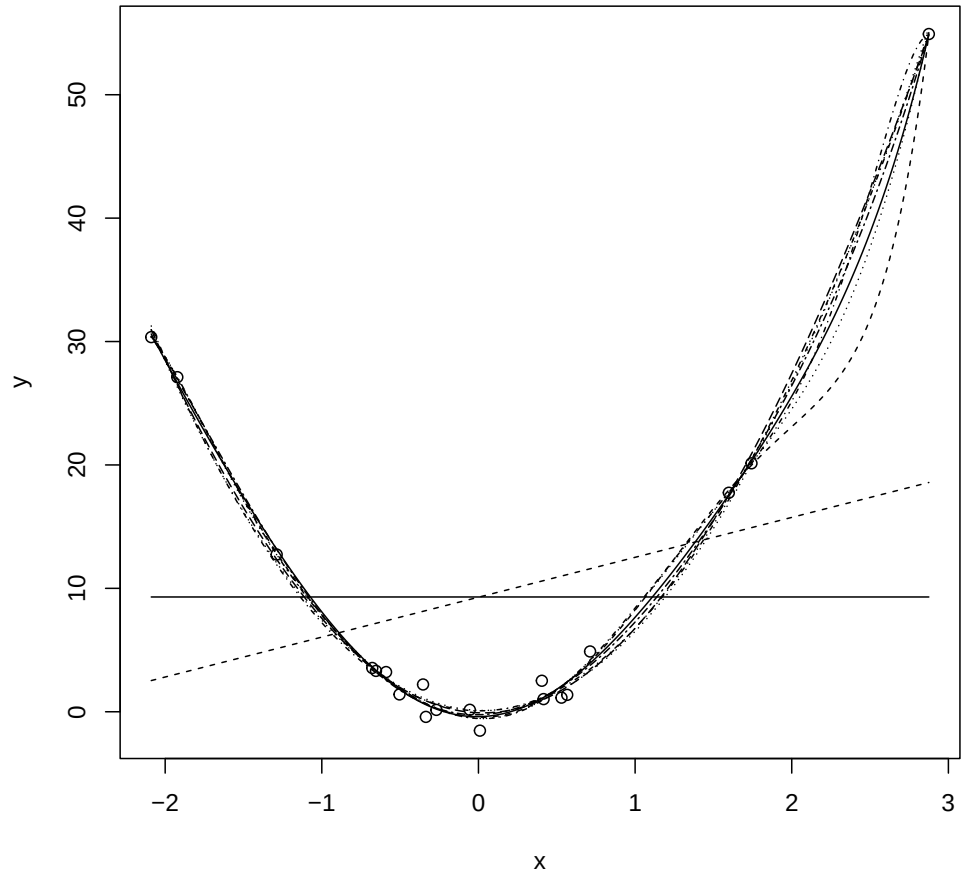

```
x = rnorm(20)
y = 7 * x^2 - 0.5 * x + rnorm(20)

plot(x, y)
curve(7 * x^2 - 0.5 * x, col = "grey", add = TRUE)
```

Figure 3.2 Scatter-plot showing sample data and the true, quadratic regression curve (grey parabola).

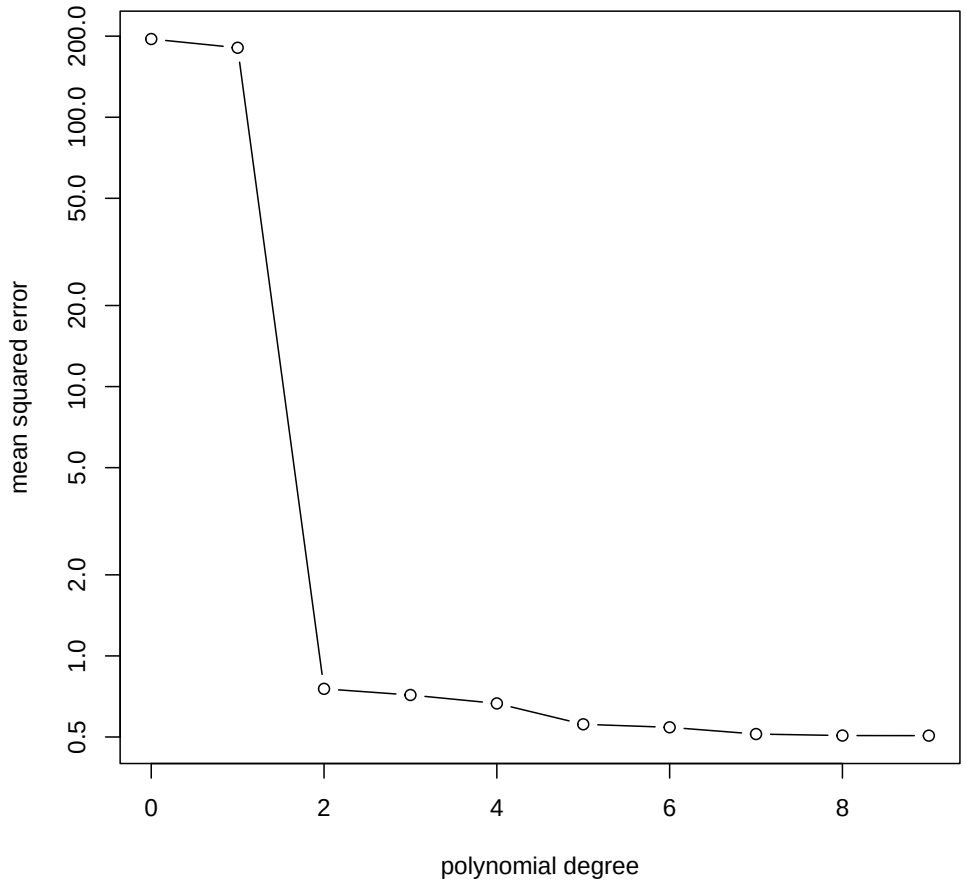
matching the noise better. In terms of the bias-variance decomposition, the bias shrinks with the model degree, but the variance of estimation grows.

Notice that the models of degrees 0 and 1 also do worse than the quadratic model — their problem is not over-fitting but *under*-fitting; they would do better




```
plot(x, y)
poly.formulae <- c("y~1", paste("y ~ poly(x,", 1:9, ")", sep = ""))
poly.formulae <- sapply(poly.formulae, as.formula)
df.plot <- data.frame(x = seq(min(x), max(x), length.out = 200))
fitted.models <- list(length = length(poly.formulae))
for (model_index in 1:length(poly.formulae)) {
    fm <- lm(formula = poly.formulae[[model_index]])
    lines(df.plot$x, predict(fm, newdata = df.plot), lty = model_index)
    fitted.models[[model_index]] <- fm
}
```

Figure 3.3 Twenty training data points (dots), and ten different fitted regression lines (polynomials of degree 0 to 9, indicated by different line types). R NOTES: The `poly` command constructs orthogonal (uncorrelated) polynomials of the specified degree from its first argument; regressing on them is conceptually equivalent to regressing on $1, x, x^2, \ldots x^{\text{degree}}$, but more numerically stable. (See `?poly`.) This builds a vector of model formulae and then fits each one in turn, storing the fitted models in a new list.




```
mse.q <- sapply(fitted.models, function(mdl) {
    mean(residuals(mdl)^2)
})
plot(0:9, mse.q, type = "b", xlab = "polynomial degree", ylab = "mean squared error",
    log = "y")
```

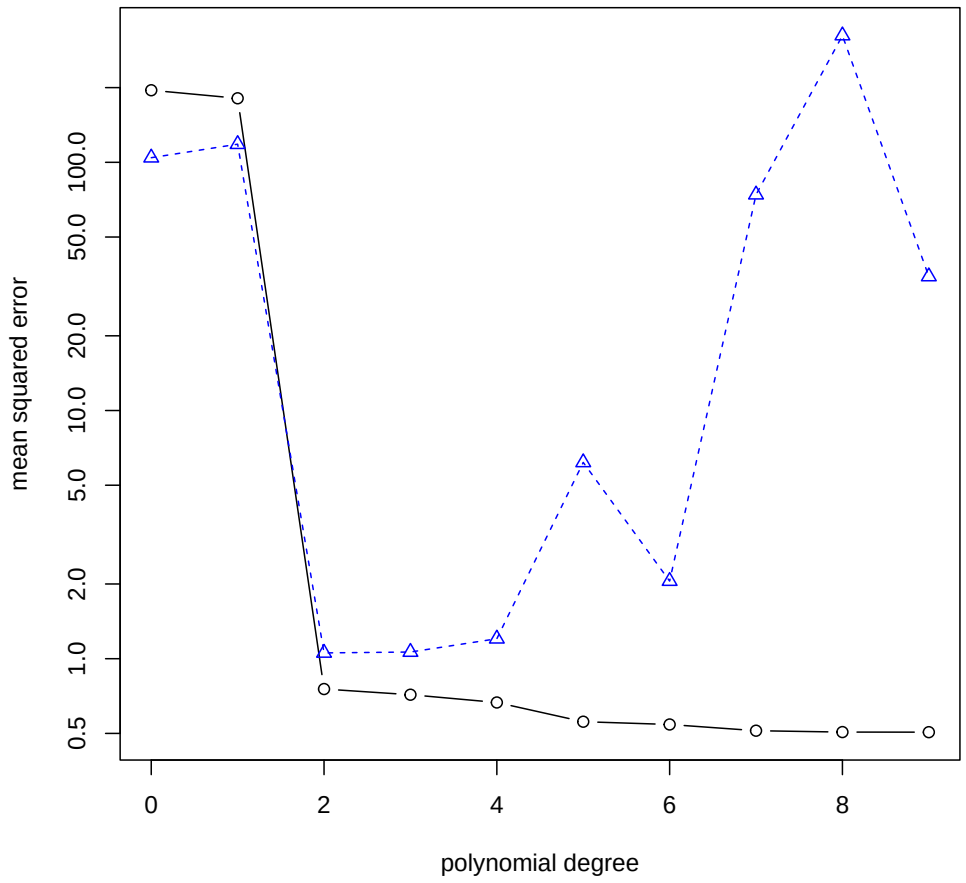
Figure 3.4 Empirical MSE vs. degree of polynomial for the data from the previous figure. Note the logarithmic scale for the vertical axis.

if they were more flexible. Plots of generalization error like this usually have a minimum. If we have a choice of models — if we need to do **model selection** — we would like to find the minimum. Even if we do not have a *choice* of models,

we might like to know how big the gap between our in-sample error and our generalization error is likely to be.

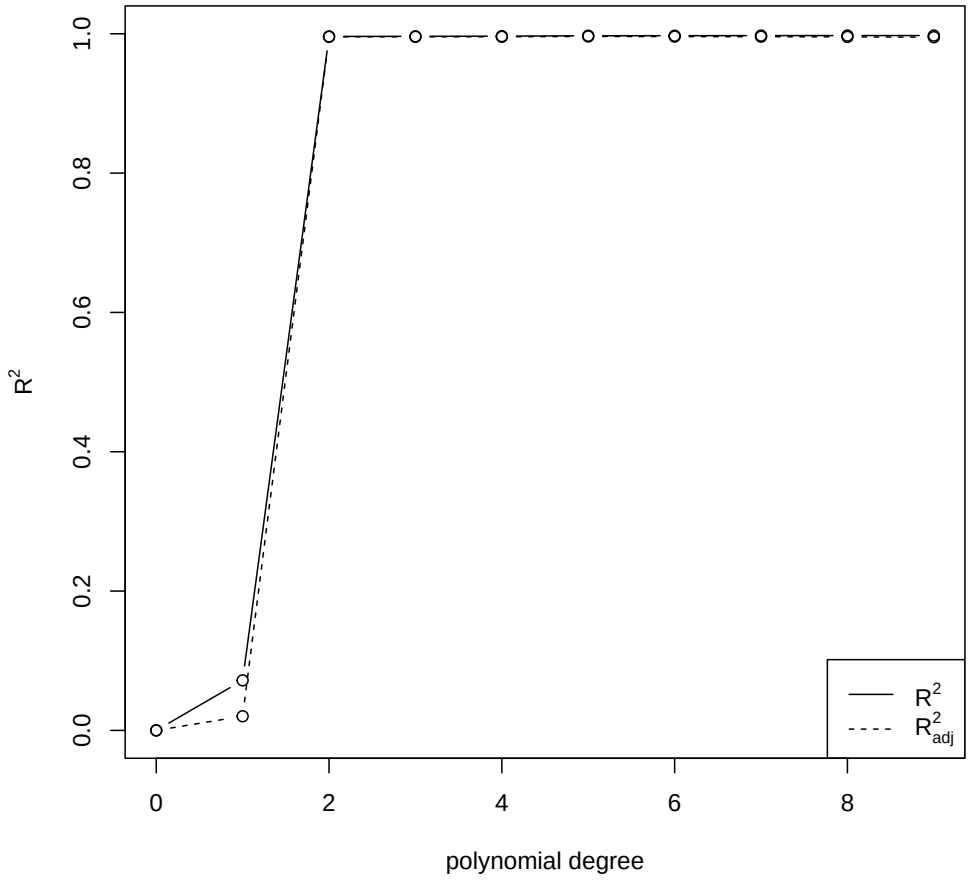
There is nothing special about polynomials here. All of the same lessons apply to variable selection in linear regression, to k-nearest neighbors (where we need to choose k), to kernel regression (where we need to choose the bandwidth), and to other methods we'll see later. In every case, there is going to be a minimum for the generalization error curve, which we'd like to find.

(A minimum with respect to what, though? In Figure 3.5, the horizontal axis is the model degree, which here is the number of parameters [minus one for the intercept]. More generally, however, what we care about is some measure of how complex the model space is, which is not necessarily the same thing as the number of parameters. What's more relevant is how flexible the class of models is, how many different functions it can approximate. Linear polynomials can approximate a smaller set of functions than quadratics can, so the latter are more complex, or have higher **capacity**. More advanced learning theory has a number of ways of quantifying this, but the details get pretty arcane, and we will just use the concept of complexity or capacity informally.)




```
x.new = rnorm(20000)
y.new = 7 * x.new^2 - 0.5 * x.new + rnorm(20000)
gmse <- function(mdl) {
    mean((y.new - predict(mdl, data.frame(x = x.new)))^2)
}
gmse.q <- sapply(fitted.models, gmse)
plot(0:9, mse.q, type = "b", xlab = "polynomial degree", ylab = "mean squared error",
    log = "y", ylim = c(min(mse.q), max(gmse.q)))
lines(0:9, gmse.q, lty = 2, col = "blue")
points(0:9, gmse.q, pch = 24, col = "blue")
```

Figure 3.5 In-sample error (black dots) compared to generalization error (blue triangles). Note the logarithmic scale for the vertical axis.




```
extract.rsqd <- function(mdl) {
    c(summary(mdl)$r.squared, summary(mdl)$adj.r.squared)
}
rsqd.q <- sapply(fitted.models, extract.rsqd)
plot(0:9, rsqd.q[1, ], type = "b", xlab = "polynomial degree", ylab = expression(R^2),
    ylim = c(0, 1))
lines(0:9, rsqd.q[2, ], type = "b", lty = "dashed")
legend("bottomright", legend = c(expression(R^2), expression(R[adj]^2)), lty = c("solid",
    "dashed"))
```

Figure 3.6 R^2 and adjusted R^2 for the polynomial fits, to reinforce §2.2.1.1's point that neither statistic is a useful measure of how well a model fits, or a good criteria for picking among models.

3.4 Cross-Validation

The most straightforward way to find the generalization error would be to do what I did above, and to use fresh, independent data from the same source — a **testing** or **validation** data-set. Call this $\mathbf{z}'_m$, as opposed to our training data $\mathbf{z}_n$. We fit our model to $\mathbf{z}_n$, and get $\widehat{\theta_n}$. The loss of this on the validation data is

$$\mathbb{E}\left[L(Z, \widehat{\theta_n})\right] + \eta'_m(\widehat{\theta_n}) \tag{3.8}$$

where now the sampling noise on the *validation* set, η'_m, is independent of $\widehat{\theta_n}$. So this gives us an unbiased estimate of the generalization error, and, if m is large, a precise one. If we need to select one model from among many, we can pick the one which does best on the validation data, with confidence that we are not just over-fitting.

The problem with this approach is that we absolutely, positively, cannot use any of the validation data in estimating the model. Since collecting data is expensive — it takes time, effort, and usually money, organization, effort and skill — this means getting a validation data set is expensive, and we often won't have that luxury.

3.4.1 Data Splitting

The next logical step, however, is to realize that we don't strictly need a separate validation set. We can just take our data and *split* it ourselves into training and testing sets. If we divide the data into two parts at random, we ensure that they have (as much as possible) the same distribution, and that they are independent of each other. Then we can act just as though we had a real validation set. Fitting to one part of the data, and evaluating on the other, gives us an unbiased estimate of generalization error. Of course it doesn't matter which half of the data is used to train and which half is used to test.

Figure 3.7 illustrates the idea with a bit of the data and linear models from §10, and Code Example 2 shows the code used to make Figure 3.7.

```
CAPA <- na.omit(read.csv("http://www.stat.cmu.edu/~cshalizi/uADA/13/hw/01/calif_penn_2011.csv"))

half_A <- sample(1:nrow(CAPA), size = nrow(CAPA)/2, replace = FALSE)
half_B <- setdiff(1:nrow(CAPA), half_A)
small_formula = "Median_house_value ~ Median_household_income"
large_formula = "Median_house_value ~ Median_household_income + Median_rooms"
small_formula <- as.formula(small_formula)
large_formula <- as.formula(large_formula)
msmall <- lm(small_formula, data = CAPA, subset = half_A)
mlarge <- lm(large_formula, data = CAPA, subset = half_A)
in.sample.mse <- function(model) {
    mean(residuals(model)^2)
}
new.sample.mse <- function(model, half) {
    test <- CAPA[half, ]
    predictions <- predict(model, newdata = test)
    return(mean((test$Median_house_value - predictions)^2))
}
```

Code Example 2: *Code used to generate the numbers in Figure 3.7.*

	Median_house_value	Median_household_income	Median_rooms
2	909600	111667	6.0
3	748700	66094	4.6
4	773600	87306	5.0
5	579200	62386	4.5
11274	209500	56667	6.0
11275	253400	71638	6.6

	Median_house_value	Median_household_income	Median_rooms
2	909600	111667	6.0
5	579200	62386	4.5

	Median_house_value	Median_household_income	Median_rooms
3	748700	66094	4.6
4	773600	87306	5.0
11274	209500	56667	6.0
11275	253400	71638	6.6

	RMSE($A \to A$)	RMSE($A \to B$)
Income only	1.6153922×10^5	1.6140141×10^5
Income + Rooms	1.2537721×10^5	1.2872457×10^5

Figure 3.7 Example of data splitting. The top table shows three columns and seven rows of the housing-price data used in §10. I then randomly split this into two equally-sized parts (next two tables). I estimate a linear model which predicts house value from income alone, and another model which predicts from income and the median number of rooms, on the first half. The third table fourth row shows the performance of each estimated model both on the first half of the data (left column) and on the second (right column). The latter is a valid estimate of generalization error. The larger model *always* has a lower in-sample error, whether or not it is really better, so the in-sample MSEs provide little *evidence* that we should use the larger model. Having a lower score under data splitting, however, *is* evidence that the larger model generalizes better. (For R commands used to get these numbers, see Code Example 2.)

```
cv.lm <- function(data, formulae, nfolds = 5) {
    data <- na.omit(data)
    formulae <- sapply(formulae, as.formula)
    n <- nrow(data)
    fold.labels <- sample(rep(1:nfolds, length.out = n))
    mses <- matrix(NA, nrow = nfolds, ncol = length(formulae))
    colnames <- as.character(formulae)
    for (fold in 1:nfolds) {
        test.rows <- which(fold.labels == fold)
        train <- data[-test.rows, ]
        test <- data[test.rows, ]
        for (form in 1:length(formulae)) {
            current.model <- lm(formula = formulae[[form]], data = train)
            predictions <- predict(current.model, newdata = test)
            test.responses <- eval(formulae[[form]][[2]], envir = test)
            test.errors <- test.responses - predictions
            mses[fold, form] <- mean(test.errors^2)
        }
    }
    return(colMeans(mses))
}
```

Code Example 3: *Function to do k-fold cross-validation on linear models, given as a vector (or list) of model formulae. Note that this only returns the CV MSE, not the parameter estimates on each fold.*

3.4.2 k-Fold Cross-Validation (CV)

The problem with data splitting is that, while it's an unbiased estimate of the risk, it is often a very noisy one. If we split the data evenly, then the test set has $n/2$ data points — we've cut in half the number of sample points we're averaging over. It would be nice if we could reduce that noise somewhat, especially if we are going to use this for model selection.

One solution to this, which is pretty much the industry standard, is what's called **k-fold cross-validation**. Pick a small integer k, usually 5 or 10, and divide the data at random into k equally-sized subsets. (The subsets are often called "folds".) Take the first subset and make it the test set; fit the models to the rest of the data, and evaluate their predictions on the test set. Now make the second subset the test set and the rest of the training sets. Repeat until each subset has been the test set. At the end, average the performance across test sets. This is the cross-validated estimate of generalization error for each model. Model selection then picks the model with the smallest estimated risk.[7] Code Example 3 performs k-fold cross-validation for linear models specified by formulae.

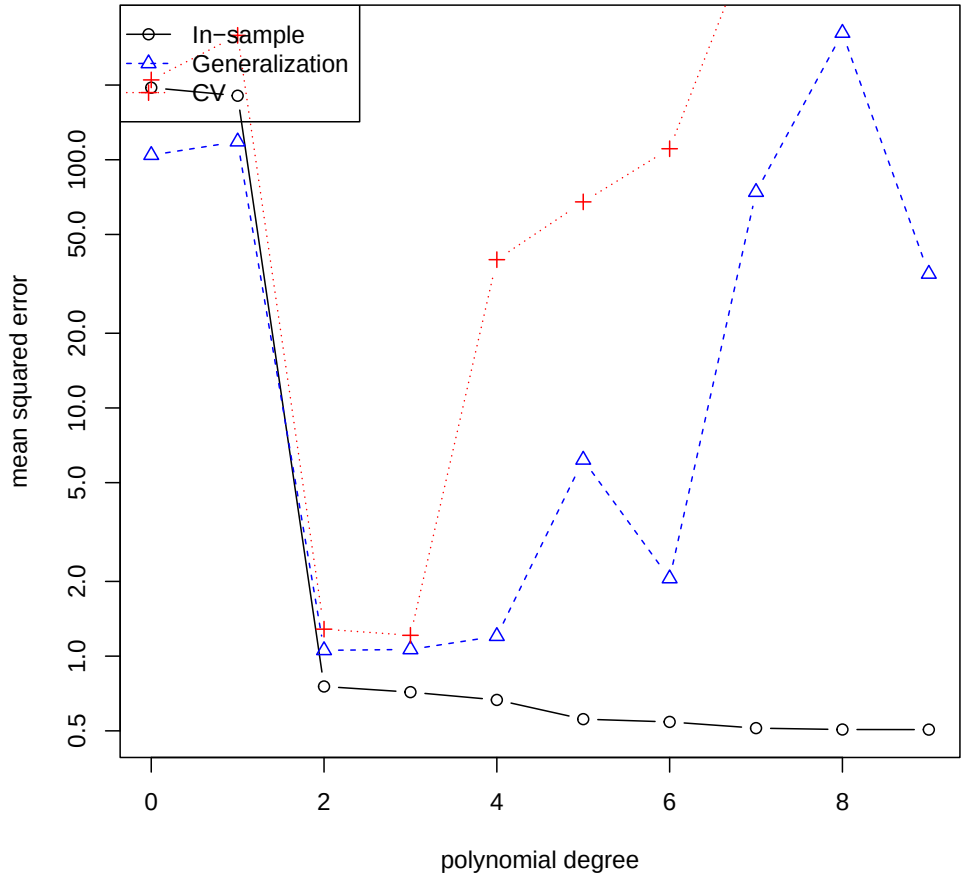
The reason cross-validation works is that it uses the existing data to simulate the process of generalizing to new data. If the full sample is large, then even the smaller portion of it in the testing data is, with high probability, fairly represen-

[7] A closely related procedure, sometimes also called "k-fold CV", is to pick $1/k$ of the data points at random to be the test set (using the rest as a training set), and then pick an *independent* $1/k$ of the data points as the test set, etc., repeating k times and averaging. The differences are subtle, but what's described in the main text makes sure that each point is used in the test set just once.

tative of the data-generating process. *Randomly* dividing the data into training and test sets makes it very unlikely that the division is rigged to favor any one model class, over and above what it would do on real new data. Of course the original data set is never *perfectly* representative of the full data, and a smaller testing set is even less representative, so this isn't ideal, but the approximation is often quite good. k-fold CV is fairly good at getting the *relative* order of different models right, that is, at controlling over-fitting.[8] Figure 3.8 demonstrates these points for the polynomial fits we considered earlier (in Figures 3.3–3.5).

Cross-validation is probably the most widely-used method for model selection, and for picking control settings, in modern statistics. There are circumstances where it can fail — especially if you give it *too many* models to pick among — but it's the first thought of seasoned practitioners, and it should be your first thought, too. The assignments to come will make you *very* familiar with it.

[8] The cross-validation score for the selected model still tends to be somewhat over-optimistic, because it's still picking the luckiest model — though the influence of luck is much attenuated. Tibshirani and Tibshirani (2009) provides a simple correction.




```
little.df <- data.frame(x = x, y = y)
cv.q <- cv.lm(little.df, poly.formulae)
plot(0:9, mse.q, type = "b", xlab = "polynomial degree", ylab = "mean squared error",
    log = "y", ylim = c(min(mse.q), max(gmse.q)))
lines(0:9, gmse.q, lty = 2, col = "blue", type = "b", pch = 2)
lines(0:9, cv.q, lty = 3, col = "red", type = "b", pch = 3)
legend("topleft", legend = c("In-sample", "Generalization", "CV"), col = c("black",
    "blue", "red"), lty = 1:3, pch = 1:3)
```

Figure 3.8 In-sample, generalization, and cross-validated MSE for the polynomial fits of Figures 3.3, 3.4 and 3.5. Note that the cross-validation is done entirely within the initial set of only 20 data points.

3.4.3 Leave-one-out Cross-Validation

Suppose we did k-fold cross-validation, but with $k = n$. Our testing sets would then consist of single points, and each point would be used in testing once. This is called **leave-one-out cross-validation**. It actually came before k-fold cross-validation, and has three advantages. First, because it estimates the performance of a model trained with $n-1$ data points, it's less biased as an estimator of the performance of a model trained with n data points than is k-fold cross-validation, which uses $\frac{k-1}{k}n$ data points. Second, leave-one-out doesn't require any random number generation, or keeping track of which data point is in which subset. Third, and more importantly, because we are only testing on *one* data point, it's often possible to find what the prediction on the left-out point would be by doing calculations on a model fit to the *whole* data. (See p. 87 below.) This means that we only have to fit each model once, rather than k times, which can be a big savings of computing time.

The drawback to leave-one-out CV is subtle but often decisive. Since each training set has $n-1$ points, any two training sets must share $n-2$ points. The models fit to those training sets tend to be strongly correlated with each other. Even though we are averaging n out-of-sample forecasts, those are correlated forecasts, so we are not really averaging away all that much noise. With k-fold CV, on the other hand, the fraction of data shared between any two training sets is just $\frac{k-2}{k-1}$, not $\frac{n-2}{n-1}$, so even though the number of terms being averaged is smaller, they are less correlated.

There are situations where this issue doesn't really matter, or where it's overwhelmed by leave-one-out's advantages in speed and simplicity, so there is certainly still a place for it, but one subordinate to k-fold CV.[9]

A Short-cut for Linear Smoothers

Suppose the model m is a linear smoother (§1.5). For each of the data points i, then, the predicted value is a linear combination of the observed values of y, $m(x_i) = \sum_j \hat{w}(x_i, x_j) y_j$ (Eq. 1.50). As in §1.5.3, define the "influence", "smoothing" or "hat" matrix $\hat{\mathbf{w}}$ by $\hat{w}_{ij} = \hat{w}(x_i, x_j)$. What happens when we hold back data point i, and then make a prediction at x_i? Well, the observed response at i can't contribute to the prediction, but otherwise the linear smoother should work as before, so

$$m^{(-i)}(x_i) = \frac{(\hat{\mathbf{w}}\mathbf{y})_i - \hat{w}_{ii} y_i}{1 - \hat{w}_{ii}} \tag{3.9}$$

[9] At this point, it may be appropriate to say a few words about the Akaike information criterion, or AIC. AIC also tries to estimate how well a model will generalize to new data. It's known that, under standard assumptions, as the sample size gets large, leave-one-out CV actually gives the same estimate as AIC for well-specified models. However, there do not seem to be any situations where AIC works where leave-one-out CV does not work at least as well. So AIC is really a very fast, but often very crude, approximation to the more accurate cross-validation. See §D.5.5.5 for more details and references.

The numerator just removes the contribution to $m(x_i)$ that came from y_i, and the denominator just re-normalizes the weights in the smoother. Now a little character-building algebra (Exercise 3.4) says that

$$y_i - m^{(-i)}(x_i) = \frac{y_i - m(x_i)}{1 - \hat{w}_{ii}} \tag{3.10}$$

The left-hand side of Eq. 3.10 is what we want to square and average to get the leave-one-out CV score, but everything on the right can be calculated from the fit we did to the whole data. The leave-one-out CV score is therefore

$$\frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - m(x_i)}{1 - \hat{w}_{ii}}\right)^2 \tag{3.11}$$

a useful result originally due to Grace Wahba (see "Further Reading" below). Thus, if we restrict ourselves to leave-one-out and to linear smoothers, we can calculate the CV score with just one estimation on the whole data, rather than n re-estimates.

An even faster calculation for leave-one-out, but only an approximate one, is also due to Wahba, who called it "generalized" cross-validation. This is just the in-sample MSE divided by $(1 - n^{-1}\operatorname{tr}\hat{\mathbf{w}})^2$. That is, rather than dividing each term in Eq. 3.11 by a unique factor that depends on its own diagonal entry in the hat matrix, we use the average of all the diagonal entries, $n^{-1}\operatorname{tr}\hat{\mathbf{w}}$. (Recall from §1.5.3.2 that $\operatorname{tr}\hat{\mathbf{w}}$ is the number of effective degrees of freedom for a linear smoother.) In addition to speed, this tends to reduce the influence of points with high values of $\hat{w}_{ii}$, which may or may not be desirable.

3.5 Warnings

Some caveats are in order.

1. All of the model-selection methods I have described, and almost all others in the literature, aim at getting models which will generalize well to new data, *if it follows the same distribution* as old data. Generalizing well even when distributions change is a much harder and much less well-understood problem (Quiñonero-Candela *et al.*, 2009). It is particularly troublesome for a lot of applications involving large numbers of human beings, because society keeps changing all the time — variables vary by definition, but the *relationships* between variables also change. (That's history.)
2. All of the standard theory of statistical inference you have learned so far presumes that you have a model which was fixed in advance of seeing the data. If you use the data to select the model, that theory becomes invalid, and it will no longer give you correct p-values for hypothesis tests, confidence sets for parameters, etc., etc. Typically, using the same data both to select a model and to do inference leads to too much confidence that the model is correct, significant, and estimated precisely.
3. All the model selection methods we have discussed aim at getting models which *predict well*. This is not necessarily the same as getting the *true theory of the world*. Presumably the true theory will also predict well, but the converse does not necessarily follow. We have seen (Fig. 1.3), and will see again (§9.2), examples of false but low-capacity models out-predicting correctly specified models at small n, because the former have such low variance of estimation.

The last two items — combining selection with inference, and parameter interpretation — deserve elaboration.

3.5.1 Inference after Selection

You have, by this point, learned a lot of inferential statistics — how to test various hypotheses, calculate p-values, find confidence regions, etc. Most likely, you have been taught procedures or calculations which all presume that the model you are working with is fixed in advance of seeing the data. But, of course, if you do model selection, the model you do inference within is *not* fixed in advance, but is actually a function of the data. What happens then?

This depends on whether you do inference with the same data used to select the model, or with another, independent data set. If it's the same data, then all of the inferential statistics become invalid — none of the calculations of probabilities on which they rest are right any more. Typically, if you select a model so that it fits the data well, what happens is that confidence regions become too small[10], as do p-values for testing hypotheses about parameters. Nothing can be trusted as it stands.

[10] Or, if you prefer, the same confidence region really has a lower confidence level, a lower probability of containing or covering the truth, than you think it does.

The essential difficulty is this: Your data are random variables. Since you're doing model selection, making your model a function of the data, that means your model is random too. That means there is some extra randomness in your estimated parameters (and everything else), which isn't accounted for by formulas which assume a fixed model (Exercise 3.5). This is not just a problem with formal model-selection devices like cross-validation. If you do an initial, exploratory data analysis before deciding which model to use — and that's generally a good idea — you are, yourself, acting as a noisy, complicated model-selection device.

There are three main approaches to this issue of **post-selection inference**.

1. *Ignore it.* This can actually make sense if you don't really care about doing inference within your selected model, you just care about what model is selected. Otherwise, I can't recommend it.

2. *Beat it with more statistical theory.* There is, as I write, a lot of interest among statisticians in working out exactly what happens to sampling distributions under various combinations of models, model-selection methods, and assumptions about the true, data-generating process. Since this is an active area of research in statistical theory, I will pass it by, with some references in §3.6.

3. *Evade it with an independent data set.* Remember that if the events A and B are probabilistically independent, then $\Pr(A|B) = \Pr(A)$. Now set $A =$ "the confidence set we calculated from this new data covers the truth" and $B =$ "the model selected from this old data was such-and-such". So long as the old and the new data are independent, it doesn't matter that the model was selected using data, rather than being fixed in advance.

The last approach is of course our old friend data splitting (§3.4.1). We divide the data into two parts, and we use one of them to select the model. We then re-estimate the selected model on the other part of the data, and only use that second part in calculating our inferential statistics. Experimentally, using part of the data to do selection, and then all of the data to do inference, does not work as well as a strict split (Faraway, 2016). Using equal amounts of data for selection and for inference is somewhat arbitrary, but, again it's not clear that there's a much better division.

Of course, if you only use a portion of your data to calculate confidence regions, they will typically be larger than if you used all of the data. (Or, if you're running hypothesis tests, fewer coefficients will be significantly different from zero, etc.) This drawback is more apparent than real, since using all of your data together to both select a model and to do inference gives you apparently-precise confidence regions which aren't actually valid.

The simple data-splitting approach to combining model selection and inference only works if the individual data points were independent to begin with. When we deal with dependent data, in Part IV, other approaches will be necessary.

3.5.2 Parameter Interpretation

In many situations, it is very natural to want to attach some substantive, real-world meaning to the parameters of our statistical model, or at least to some of them. I have mentioned examples above like astronomy, and it is easy to come up with many others from the natural sciences. This is also extremely common in the social sciences. It is fair to say that this is much less carefully attended to than it should be.

To take just one example, consider the paper "Luther and Suleyman" by Prof. Murat Iyigun (Iyigun, 2008). The major idea of the paper is to try to help explain why the Protestant Reformation was not wiped out during the European wars of religion (or alternately, why the Protestants did not crush all the Catholic powers), leading western Europe to have a mixture of religions, with profound consequences. Iyigun's contention is that the European Christians were so busy fighting the Ottoman Turks, or perhaps so afraid of what might happen if they did not, that conflicts among the Europeans were suppressed. To quote his abstract:

> at the turn of the sixteenth century, Ottoman conquests lowered the number of all newly initiated conflicts among the Europeans roughly by 25 percent, while they dampened all longer-running feuds by more than 15 percent. The Ottomans' military activities influenced the length of intra-European feuds too, with each Ottoman-European military engagement shortening the duration of intra-European conflicts by more than 50 percent.

To back this up, and provide those quantitative figures, Prof. Iyigun estimates linear regression models, of the form[11]

$$Y_t = \beta_0 + \beta_1 X_t + \beta_2 Z_t + \beta_3 U_t + \epsilon_t \tag{3.12}$$

where Y_t is "the number of violent conflicts initiated among or within continental European countries at time t"[12], X_t is "the number of conflicts in which the Ottoman Empire confronted European powers at time t", Z_t is "the count at time t of the newly initiated number of Ottoman conflicts with others and its own domestic civil discords", U_t is control variables reflecting things like the availability of harvests to feed armies, and ϵ_t is Gaussian noise.

The qualitative idea here, about the influence of the Ottoman Empire on the European wars of religion, has been suggested by quite a few historians before[13]. The point of this paper is to support this rigorously, and make it precise. That support and precision requires Eq. 3.12 to be an accurate depiction of at least part of the process which led European powers to fight wars of religion. Prof. Iyigun, after all, wants to be able to interpret a negative estimate of β_1 as saying that fighting off the Ottomans *kept* Christians from fighting each other. If Eq. 3.12 is inaccurate, if the model is badly mis-specified, however, β_1 becomes the best approximation to the truth within a systematically wrong model, and the support for claims like "Ottoman conquests lowered the number of all newly initiated conflicts among the Europeans roughly by 25 percent" drains away.

[11] His Eq. 1 on pp. 1473; I have modified the notation to match mine.
[12] In one part of the paper; he uses other dependent variables elsewhere.
[13] See §1–2 of Iyigun (2008), and MacCulloch (2004, *passim*).

To back up the use of Eq. 3.12, Prof. Iyigun looks at a range of slightly different linear-model specifications (e.g., regress the number of intra-Christian conflicts in year t on the number of Ottoman attacks in year $t-1$), and slightly different methods of estimating the parameters. What he does *not* do is look at the other implications of the model: that residuals should be (at least approximately) Gaussian, that they should be unpredictable from the regressor variables. He does not look at whether the relationships he thinks are linear really are linear (see Chapters 4, 8, and 9). He does not try to simulate his model and look at whether the patterns of European wars it produces resemble actual history (see Chapter 5). He does not try to check whether he has a model which really supports causal inference, though he has a causal question (see Part III).

I do not say any of this to denigrate Prof. Iyigun. His paper is actually *much better* than most quantitative work in the social sciences. This is reflected by the fact that it was published in the *Quarterly Journal of Economics*, one of the most prestigious, and rigorously-reviewed, journals in the field. The point is that by the end of this course, you will have the tools to do *better*.

3.6 Further Reading

Data splitting and cross-validation go back in statistical practice for many decades, though often as a very informal tool. One of the first important papers on the subject was Stone (1974), which goes over the earlier history. Arlot and Celisse (2010) is a good recent review of cross-validation. Faraway (1992, 2016) reviews computational evidence that data splitting reduces the over-confidence that results from model selection even if one only wants to do prediction. Györfi *et al.* (2002, chs. 7–8) has important results on data splitting and cross-validation, though the proofs are much more advanced than this book.

Some comparatively easy starting points on statistical learning theory are Kearns and Vazirani (1994), Cristianini and Shawe-Taylor (2000) and Mohri *et al.* (2012). At a more advanced level, look at the tutorial papers by Bousquet *et al.* (2004); von Luxburg and Schölkopf (2008), or the textbooks by Vidyasagar (2003) and by Anthony and Bartlett (1999) (the latter is much more general than its title suggests), or read the book by Vapnik (2000) (one of the founders). Hastie *et al.* (2009), while invaluable, is much more oriented towards models and practical methods than towards learning *theory*.

On model selection in general, the best recent summary is the book by Claeskens and Hjort (2008); it is more theoretically demanding than this book, but includes many real-data examples.

The literature on doing statistical inference after model selection by accounting for selection effects, rather than simple data splitting, is already large and rapidly growing. Taylor and Tibshirani (2015) is a comparatively readable introduction to the "selective inference" approach associated with those authors and their collaborators. Tibshirani *et al.* (2015) draws connections between this approach and the bootstrap (ch. 6). Berk *et al.* (2013) provides yet another approach to post-selection inference; nor is this an exhaustive list. For adaptations of data splitting to dependent data, see Wang (2016) (for networks) and Lunde (2018) (for time series).

White (1994) is a thorough treatment of parameter estimation in models which may be mis-specified, and some general tests for mis-specification. It also briefly discusses the interpretation of parameters in mis-specified models. That topic deserves a more in-depth treatment, but I don't know of a really good one.

Exercises

3.1 Suppose that one of our model classes contains the true and correct model, but we also consider more complicated and flexible model classes. Does the bias-variance trade-off mean that we will over-shoot the true model, and always go for something more flexible, when we have enough data? (This would mean there was such a thing as *too much* data to be reliable.)

3.2 Derive the formula for the generalization risk in the situation depicted in Figure 3.1, as given by the `true.risk` function in the code for that figure. In particular, explain to yourself where the constants 0.5^2 and $1/12$ come from.

3.3 *"Optimism" and degrees of freedom* Suppose we get data of the form $Y_i = \mu(x_i) + \epsilon_i$,

where the noise terms ϵ_i have mean zero, are uncorrelated, and all have variance σ^2. We use a linear smoother (§1.5) to estimate $\widehat{\mu}$ from n such data points. The **optimism** of the estimate[14] is

$$\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{\mu}(x_i))^2\right] - \mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{\mu}(x_i))^2\right] \tag{3.13}$$

where Y_i' is an independent copy of Y_i. That is, the optimism is the difference between the in-sample MSE, and how well the model would predict on new data taken at exactly the same x_i values.

1. Find a formula for the optimism in terms of n, σ^2, and the number of effective degrees of freedom (in the sense of §1.5.3.2).
2. When (and why) does $\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i' - \widehat{\mu}(x_i))^2\right]$ differ from the risk?

(Cf. §D.5.4.3.)

3.4 Derive Wahba's short-cut formula for leave-one-out cross-validation of linear smoothers by first deriving Eq. 3.9, then Eq. 3.10, and finally using them to get Eq. 3.11.

3.5 *The perils of post-selection inference, and data splitting to the rescue*[15] Generate a 1000×101 array, where all the entries are IID standard Gaussian variables. We'll call the first column the response variable Y, and the others the predictors $X_1, \ldots X_{100}$. By design, there is no true relationship between the response and the predictors (but all the usual linear-Gaussian-modeling assumptions hold).

1. Estimate the model $Y = \beta_0 + \beta_1 X_1 + \beta_{50} X_{50} + \epsilon$. Extract the p-value for the F test of the whole model. Repeat the simulation, estimation and testing 100 times, and plot the histogram of the p-values. What does it look like? What should it look like?
2. Use the `step` function to select a linear model by forward stepwise selection. Extract the p-value for the F-test of the selected model. Repeat 100 times and plot the histogram of p-values. Explain what's going on.
3. Again use `step` to select a model based on one random 1000×101 array. Now re-estimate the selected model on a new 1000×101 array, and extract the new p-value. Repeat 100 times, with new selection and inference sets each time, and plot the histogram of p-values.

[14] The term was apparently introduced by Efron (1986), which also derived the result, though much shorter proofs have been found since then.

[15] Inspired by Freedman (1983).

4

Using Nonparametric Smoothing in Regression

Having spent long enough running down linear regression, and thought through evaluating predictive models, it is time to turn to constructive alternatives, which are (also) based on smoothing.

Recall the basic kind of smoothing we are interested in: we have a response variable Y, some input variables which we bind up into a vector X, and a collection of data values, $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$. By "smoothing", I mean that predictions are going to be weighted averages of the observed responses in the training data:

$$\widehat{\mu}(x) = \sum_{i=1}^{n} y_i w(x, x_i, h) \tag{4.1}$$

Most smoothing methods have a control setting, here written h, that says *how much* to smooth. With k nearest neighbors, for instance, the weights are $1/k$ if x_i is one of the k-nearest points to x, and $w = 0$ otherwise, so large k means that each prediction is an average over many training points. Similarly with kernel regression, where the degree of smoothing is controlled by the bandwidth.

Why do we want to do this? How do we pick how much smoothing to do?

4.1 How Much Should We Smooth?

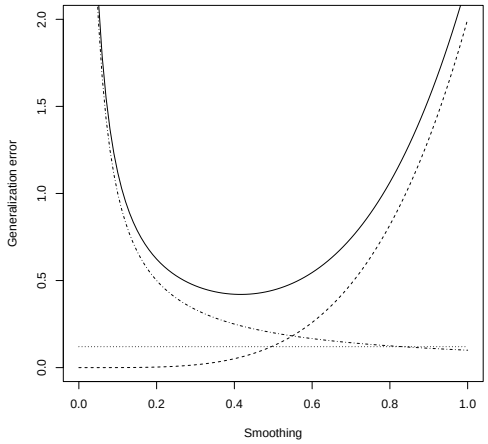
When we smooth very little ($h \to 0$), then we can match very small, fine-grained or sharp aspects of the true regression function, if there are such. That is, less smoothing leads to less bias. At the same time, less smoothing means that each of our predictions is going to be an average over (in effect) fewer observations, making the prediction noisier. Smoothing less increases the variance of our estimate. Since

$$\text{(total error)} = \text{(noise)} + \text{(bias)}^2 + \text{(variance)} \tag{4.2}$$

(Eq. 1.28), if we plot the different components of error as a function of h, we typically get something that looks like Figure 4.1. Because changing the amount of smoothing has opposite effects on the bias and the variance, there is an optimal amount of smoothing, where we can't reduce one source of error without increasing the other. We therefore want to find that optimal amount of smoothing, which is where cross-validation comes in.

You should note, at this point, that the optimal amount of smoothing depends

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/



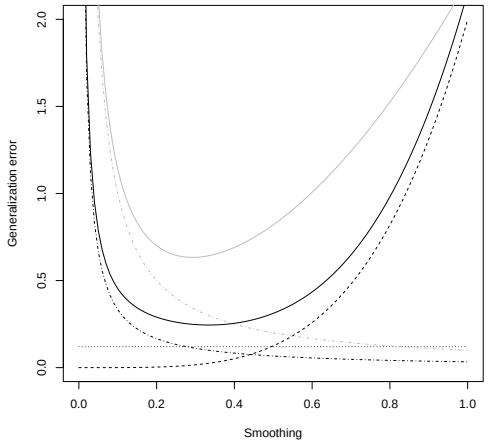

```
curve(2 * x^4, from = 0, to = 1, lty = 2, xlab = "Smoothing", ylab = "Generalization error")
curve(0.12 + x - x, lty = 3, add = TRUE)
curve(1/(10 * x), lty = 4, add = TRUE)
curve(0.12 + 2 * x^4 + 1/(10 * x), add = TRUE)
```

Figure 4.1 Decomposition of the generalization error of smoothing: the total error (solid) equals process noise (dotted) plus approximation error from smoothing (=squared bias, dashed) and estimation variance (dot-and-dash). The numerical values here are arbitrary, but the functional forms (squared bias $\propto h^4$, variance $\propto n^{-1}h^{-1}$) are representative of kernel regression (Eq. 4.12).

on the real regression curve, on our smoothing method, *and* on how much data we have. This is because the variance contribution generally shrinks as we get more data.[1] If we get more data, we go from Figure 4.1 to Figure 4.2. The minimum of the over-all error curve has shifted to the left, and we should smooth less.

Strictly speaking, **parameters** are properties of the data-generating process alone, so the optimal amount of smoothing is not really a parameter. If you do think of it as a parameter, you have the problem of why the "true" value changes as you get more data. It's better thought of as a setting or **control variable** in the smoothing method, to be adjusted as convenient.

[1] Sometimes bias changes as well. Noise does not (why?).




```
curve(2 * x^4, from = 0, to = 1, lty = 2, xlab = "Smoothing", ylab = "Generalization error")
curve(0.12 + x - x, lty = 3, add = TRUE)
curve(1/(10 * x), lty = 4, add = TRUE, col = "grey")
curve(0.12 + 2 * x^2 + 1/(10 * x), add = TRUE, col = "grey")
curve(1/(30 * x), lty = 4, add = TRUE)
curve(0.12 + 2 * x^4 + 1/(30 * x), add = TRUE)
```

Figure 4.2 Consequences of adding more data to the components of error: noise (dotted) and bias (dashed) don't change, but the new variance curve (dotted and dashed, black) is to the left of the old (greyed), so the new over-all error curve (solid black) is lower, and has its minimum at a smaller amount of smoothing than the old (solid grey).

4.2 Adapting to Unknown Roughness

Figure 4.3, which graphs two functions, r and s. Both are "smooth" functions in the mathematical sense[2]. We could Taylor-expand both functions to approximate their values anywhere, just from knowing enough derivatives at one point x_0.[3] If instead of knowing the derivatives at x_0 we have the values of the functions at a sequence of points $x_1, x_2, \ldots x_n$, we could use interpolation to fill out the rest of the curve. Quantitatively, however, r is less smooth than s — it changes much more rapidly, with many reversals of direction. For the same degree of accuracy in the interpolation r needs more, and more closely spaced, training points x_i than does s.

Now suppose that we don't get to actually get to see r and s, but rather just $r(x)+\epsilon$ and $s(x)+\eta$, for various x, where ϵ and η are noise. (To keep things simple I'll assume they're constant-variance, IID Gaussian noises, say with $\sigma = 0.15$.) The data now look something like Figure 4.4. Can we recover the curves?

As remarked in Chapter 1, if we had many measurements at the same x, then we could find the expectation value by averaging: the regression function $\mu(x) = \mathbb{E}[Y|X = x]$, so with multiple observations $x_i = x$, the mean of the corresponding y_i would (by the law of large numbers) converge on $\mu(x)$. Generally, however, we have at most one measurement per value of x, so simple averaging won't work. Even if we just confine ourselves to the x_i where we have observations, the mean-squared error would always be σ^2, the noise variance. However, our estimate would be unbiased.

Smoothing methods try to use multiple measurements at points x_i which are *near* the point of interest x. If the regression function is smooth, as we're assuming it is, $\mu(x_i)$ will be close to $\mu(x)$. Remember that the mean-squared error is the sum of bias (squared) and variance. Averaging values at $x_i \neq x$ is going to introduce bias, but averaging independent terms together also reduces variance. If smoothing gets rid of more variance than it adds bias, we come out ahead.

Here's a little math to see it. Let's assume that we can do a first-order Taylor expansion (Figure B.1), so

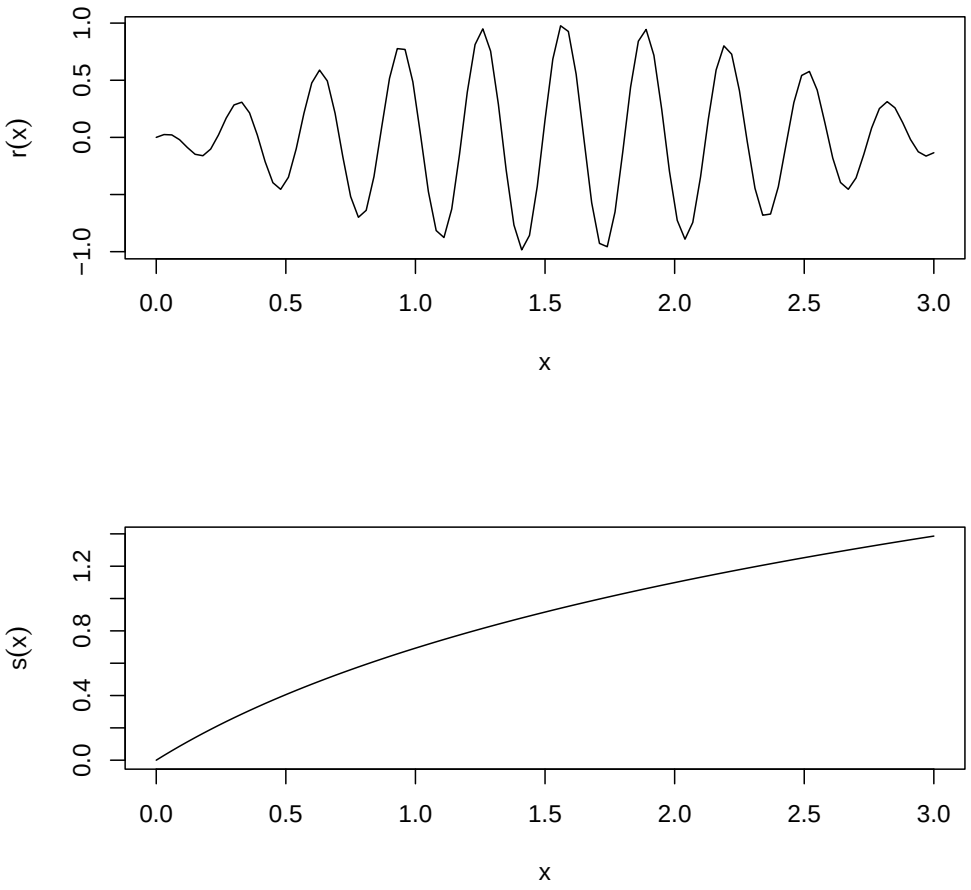
$$\mu(x_i) \approx \mu(x) + (x_i - x)\mu'(x) \tag{4.3}$$

and

$$y_i \approx \mu(x) + (x_i - x)\mu'(x) + \epsilon_i \tag{4.4}$$

Now we average: to keep the notation simple, abbreviate the weight $w(x_i, x, h)$

[2] They are "C^∞": continuous, with continuous derivatives to all orders.
[3] See App. B for a refresher on Taylor expansions.




```
par(mfcol = c(2, 1))
true.r <- function(x) {
    sin(x) * cos(20 * x)
}
true.s <- function(x) {
    log(x + 1)
}
curve(true.r(x), from = 0, to = 3, xlab = "x", ylab = expression(r(x)))
curve(true.s(x), from = 0, to = 3, xlab = "x", ylab = expression(s(x)))
par(mfcol = c(1, 1))
```

Figure 4.3 Two curves for the running example. Above, $r(x) = \sin x \cos 20x$; below, $s(x) = \log 1 + x$ (we will not use this information about the exact functional forms).

by just w_i.

$$\widehat{\mu}(x) = \sum_{i=1}^{n} y_i w_i \tag{4.5}$$

$$= \sum_{i=1}^{n} (\mu(x) + (x_i - x)\mu'(x) + \epsilon_i) w_i \tag{4.6}$$

$$= \mu(x) + \sum_{i=1}^{n} w_i \epsilon_i + \mu'(x) \sum_{i=1}^{n} w_i (x_i - x) \tag{4.7}$$

$$\widehat{\mu}(x) - \mu(x) = \sum_{i=1}^{n} w_i \epsilon_i + \mu'(x) \sum_{i=1}^{n} w_i (x_i - x) \tag{4.8}$$

$$\mathbb{E}\left[(\widehat{\mu}(x) - \mu(x))^2\right] = \sigma^2 \sum_{i=1}^{n} w_i^2 + \mathbb{E}\left[\left(\mu'(x) \sum_{i=1}^{n} w_i (x_i - x)\right)^2\right] \tag{4.9}$$

(Remember that: $\sum w_i = 1$; $\mathbb{E}[\epsilon_i] = 0$; ϵ is uncorrelated with everything; and $\mathbb{V}[\epsilon_i] = \sigma^2$.)

The first term on the final right-hand side is an estimation variance, which will tend to shrink as n grows. (If we just did a simple global mean, $w_i = 1/n$ for all i, so we'd get σ^2/n, just like in baby stats.) The second term, an expectation, is bias, which grows as x_i gets further from x, and as the magnitudes of the derivatives grow, i.e., this term's growth varies with *how* smooth or wiggly the regression function is. For smoothing to work, w_i had better shrink as $x_i - x$ and $\mu'(x)$ grow.[4] Finally, all else being equal, w_i should also shrink with n, so that the over-all size of the sum shrinks as we get more data.

To illustrate, let's try to estimate $r(1.6)$ and $s(1.6)$ from the noisy observations. We'll try a simple approach, just averaging all values of $r(x_i) + \epsilon_i$ and $s(x_i) + \eta_i$ for $1.5 < x_i < 1.7$ with equal weights. For r, this gives 0.54, while $r(1.6) = 0.83$. For g, this gives 0.94, with $s(1.6) = 0.96$. (See figure 4.5.) The same window size creates a much larger bias with the rougher, more rapidly changing r than with the smoother, more slowly changing s. Varying the size of the averaging window will change the amount of error, and it will change it in different ways for the two functions.

If one does a more careful second-order Taylor expansion like that leading to Eq. 4.9, specifically for kernel regression, one can show that the bias at x is

$$\mathbb{E}\left[\widehat{\mu}(x) - \mu(x) | X_1 = x_1, \ldots X_n = x_n\right] = h^2 \left[\frac{1}{2}\mu''(x) + \frac{\mu'(x) f'(x)}{f(x)}\right] \sigma_K^2 + o(h^2) \tag{4.10}$$

where f is the density of x, and $\sigma_K^2 = \int u^2 K(u) du$, the variance of the probability density corresponding to the kernel[5]. The μ'' term just comes from the second-

[4] The higher derivatives of μ also matter, since we should really keep more than just the first term in the Taylor expansion. The details get messy, but Eq. 4.12 below gives the upshot for kernel smoothing.

[5] If you are not familiar with the "order" symbols O and o, see Appendix A.

order part of the Taylor expansion. To see where the $\mu' f'$ term comes from, imagine first that x is a mode of the distribution, so $f'(x) = 0$. As h shrinks, only training points where X_i is very close to x will have any weight in $\widehat{\mu}(x)$, and their distribution will be roughly symmetric around x (at least once h is sufficiently small). So, at mode, $\mathbb{E}\left[w(X_i, x, h)(X_i - x)\widehat{\mu}(x)\right] \approx 0$. Away from a mode, there will tend to be more training points on one side or the other of x, depending on the sign of $f'(x)$, and this induces a bias. The tricky part of the analysis is concluding that the bias has exactly the form given above.[6]

One can also work out the variance of the kernel regression estimate,

$$\mathbb{V}\left[\widehat{\mu}(x)|X_1 = x_1, \ldots X_n = x_n\right] = \frac{\sigma^2(x)R(K)}{nhf(x)} + o((nh)^{-1}) \tag{4.11}$$

where $R(K) \equiv \int K^2(u)du$. Roughly speaking, the width of the region where the kernel puts non-trivial weight is about h, so there will be about $nhf(x)$ training points available to estimate $\widehat{\mu}(x)$. Each of these has a y_i value, equal to $\mu(x)$ plus noise of variance $\sigma^2(x)$. The final factor of $R(K)$ accounts for the average weight.

Putting the bias together with the variance, we get an expression for the mean squared error of the kernel regression at x:

$$MSE(x) = \sigma^2(x) + h^4\left[\frac{1}{2}\mu''(x) + \frac{\mu'(x)f'(x)}{f(x)}\right]^2(\sigma_K^2)^2 + \frac{\sigma^2(x)R(K)}{nhf(x)} + o(h^4) + o((nh)^{-1}) \tag{4.12}$$

Eq. 4.12 tells us that, in principle, there is a single optimal choice of bandwidth h, an optimal degree of smoothing. We could find it by taking Eq. 4.12, differentiating with respect to the bandwidth, and setting everything to zero (neglecting the o terms):

$$0 = 4h^3\left[\frac{1}{2}\mu''(x) + \frac{\mu'(x)f'(x)}{f(x)}\right]^2(\sigma_K^2)^2 - \frac{\sigma^2(x)R(K)}{nh^2 f(x)} \tag{4.13}$$

$$h = \left(n\frac{4f(x)(\sigma_K^2)^2\left[\frac{1}{2}\mu''(x) + \frac{\mu'(x)f'(x)}{f(x)}\right]^2}{\sigma^2(x)R(K)}\right)^{-1/5} \tag{4.14}$$

Of course, this expression for the optimal h involves the unknown derivatives $\mu'(x)$ and $\mu''(x)$, plus the unknown density $f(x)$ and its unknown derivative $f'(x)$. But if we knew the derivative of the regression function, we would basically know the function itself (just integrate), so we seem to be in a vicious circle, where we need to know the function before we can learn it.[7]

One way of expressing this is to talk about how well a smoothing procedure

[6] Exercise 4.1 sketches the demonstration for the special case of the uniform ("boxcar") kernel.

[7] You may be wondering why I keep talking about *the* optimal bandwidth, when Eq. 4.14 makes it seem that the bandwidth should vary with x. One can go through pretty much the same sort of analysis in terms of the *expected* values of the derivatives, and the qualitative conclusions will be the same, but the notational overhead is even worse. Alternatively, there are techniques for variable-bandwidth smoothing.

would work, if an Oracle were to tell us the derivatives, or (to cut to the chase) the optimal bandwidth h_{opt}. Since most of us do not have access to such oracles, we need to *estimate* h_{opt}. Once we have this estimate, $\widehat{h}$, then we get our weights and our predictions, and so a certain mean-squared error. Basically, our MSE will be the Oracle's MSE, plus an extra term which depends on how far $\widehat{h}$ is to h_{opt}, and how sensitive the smoother is to the choice of bandwidth.

What would be really nice would be an **adaptive** procedure, one where our actual MSE, using $\widehat{h}$, approaches the Oracle's MSE, which it gets from h_{opt}. This would mean that, in effect, we are *figuring out* how rough the underlying regression function is, and so how much smoothing to do, rather than having to guess or be told. An adaptive procedure, if we can find one, is a partial[8] substitute for prior knowledge.

4.2.1 Bandwidth Selection by Cross-Validation

The most straight-forward way to pick a bandwidth, and one which generally manages to be adaptive, is in fact cross-validation; k-fold CV is usually somewhat better than leave-one-out, but the latter often works acceptably too. The usual procedure is to come up with an initial grid of candidate bandwidths, and then use cross-validation to estimate how well each one of them would generalize. The one with the lowest error under cross-validation is then used to fit the regression curve to the whole data[9].

Code Example 4 shows how it would work in R, with a one predictor variable, borrowing the `npreg` function from the `np` library (Hayfield and Racine, 2008).[10] The return value has three parts. The first is the actual best bandwidth. The second is a vector which gives the cross-validated mean-squared errors of all the different bandwidths in the vector `bandwidths`. The third component is an array which gives the MSE for each bandwidth on each fold. It can be useful to know things like whether the difference between the CV score of the best bandwidth and the runner-up is bigger than their fold-to-fold variability.

Figure 4.7 plots the CV estimate of the (root) mean-squared error versus bandwidth for our two curves. Figure 4.8 shows the data, the actual regression functions and the estimated curves with the CV-selected bandwidths. This illustrates *why* picking the bandwidth by cross-validation works: the curve of CV error against bandwidth is actually a pretty good approximation to the true curve of generalization error (which would look like Figure 4.1), so optimizing the CV error is close to optimizing the generalization error.

Notice, by the way, in Figure 4.7, that the rougher curve is more sensitive to the choice of bandwidth, and that the smoother curve always has a lower

[8] Only partial, because we'd *always* do better if the Oracle would just tell us h_{opt}.

[9] Since the optimal bandwidth is $\propto n^{-1/5}$, and the training sets in cross-validation are smaller than the whole data set, one might adjust the bandwidth proportionally. However, if n is small enough that this makes a big difference, the sheer noise in bandwidth estimation usually overwhelms this.

[10] The package has methods for automatically selecting bandwidth by cross-validation — see §4.6 below.

```
cv_bws_npreg <- function(x, y, bandwidths = (1:50)/50, nfolds = 10) {
    require(np)
    n <- length(x)
    stopifnot(n > 1, length(y) == n)
    stopifnot(length(bandwidths) > 1)
    stopifnot(nfolds > 0, nfolds == trunc(nfolds))

    fold_MSEs <- matrix(0, nrow = nfolds, ncol = length(bandwidths))
    colnames(fold_MSEs) = bandwidths

    case.folds <- sample(rep(1:nfolds, length.out = n))
    for (fold in 1:nfolds) {
        train.rows = which(case.folds != fold)
        x.train = x[train.rows]
        y.train = y[train.rows]
        x.test = x[-train.rows]
        y.test = y[-train.rows]
        for (bw in bandwidths) {
            fit <- npreg(txdat = x.train, tydat = y.train, exdat = x.test, eydat = y.test,
                bws = bw)
            fold_MSEs[fold, paste(bw)] <- fit$MSE
        }
    }
    CV_MSEs = colMeans(fold_MSEs)
    best.bw = bandwidths[which.min(CV_MSEs)]
    return(list(best.bw = best.bw, CV_MSEs = CV_MSEs, fold_MSEs = fold_MSEs))
}
```

CODE EXAMPLE 4: *Cross-validation for univariate kernel regression. The* `colnames` *trick: component names have to be character strings; other data types will be coerced into characters when we assign them to be names. Later, when we want to refer to a bandwidth column by its name, we wrap the name in another coercing function, such as* `paste`*. — The is* just *demo of how cross-validation for bandwidth selection works in principle;* don't *use it blindly on data, or in assignments. (That goes double for the vector of default bandwidths.)*

mean-squared error. Also notice that, at the minimum, one of the cross-validation estimates of generalization error is smaller than the true system noise level; this shows that cross-validation doesn't completely correct for optimism[11].

We still need to come up with an initial set of candidate bandwidths. For reasons which will drop out of the math in Chapter 14, it's often reasonable to start around $1.06 s_X / n^{1/5}$, where s_X is the sample standard deviation of X. However, it is hard to be very precise about this, and good results often require some honest trial and error.

4.2.2 Convergence of Kernel Smoothing and Bandwidth Scaling

Go back to Eq. 4.12 for the mean squared error of kernel regression. As we said, it involves some unknown constants, but we can bury them inside big-O order

[11] Tibshirani and Tibshirani (2009) gives a fairly straightforward way to adjust the estimate of the generalization error for the selected model or bandwidth, but that doesn't influence the choice of the best bandwidth.

symbols, which also absorb the little-o remainder terms:

$$MSE(h) = \sigma^2(x) + O(h^4) + O((nh)^{-1}) \tag{4.15}$$

The $\sigma^2(x)$ term is going to be there no matter what, so let's look at the excess risk over and above the intrinsic noise:

$$MSE(h) - \sigma^2(x) = O(h^4) + O((nh)^{-1}) \tag{4.16}$$

That is, the (squared) bias from the kernel's only approximately getting the curve is proportional to the fourth power of the bandwidth, but the variance is inversely proportional to the product of sample size and bandwidth. If we kept h constant and just let $n \to \infty$, we'd get rid of the variance, but we'd be left with the bias. To get the MSE to go to zero, we need to let the bandwidth h change with n — call it h_n. Specifically, suppose $h_n \to 0$ as $n \to \infty$, but $nh_n \to \infty$. Then, by Eq. 4.16, the risk (generalization error) of kernel smoothing is approaching that of the ideal predictor.

What is the best bandwidth? We saw in Eq. 4.14 that it is (up to constants)

$$h_{\text{opt}} = O(n^{-1/5}) \tag{4.17}$$

If we put this bandwidth into Eq. 4.16, we get

$$MSE(h) - \sigma^2(x) = O\left(\left(n^{-1/5}\right)^4\right) + O\left(n^{-1}\left(n^{-1/5}\right)^{-1}\right) = O\left(n^{-4/5}\right) + O\left(n^{-4/5}\right) = O\left(n^{-4/5}\right) \tag{4.18}$$

That is, the excess prediction error of kernel smoothing over and above the system noise goes to zero as $1/n^{0.8}$. Notice, by the way, that the contributions of bias and variance to the generalization error are both of the same order, $n^{-0.8}$.

Is this fast or slow? We can compare it to what would happen with a parametric model, say with parameter θ. (For linear regression, θ would be the vector of slopes and the intercept.) The optimal value of the parameter, θ_0, minimizes the mean-squared error. At θ_0, the parametric model has MSE

$$MSE(\theta_0) = \sigma^2(x) + b(x, \theta_0) \tag{4.19}$$

where b is the bias of the parametric model; this is zero when the parametric model is true[12]. Since θ_0 is unknown and must be estimated, one typically has $\widehat{\theta} - \theta_0 = O(1/\sqrt{n})$. Because the error is minimized at θ_0, the first derivatives of MSE at θ_0 are 0. Doing a second-order Taylor expansion of the parametric model contributes an error $O((\widehat{\theta} - \theta_0)^2)$, so altogether

$$MSE(\widehat{\theta}) - \sigma^2(x) = b(x, \theta_0) + O(1/n) \tag{4.20}$$

This means parametric models converge more quickly (n^{-1} goes to zero faster than $n^{-0.8}$), but they typically converge to the wrong answer ($b^2 > 0$). Kernel smoothing converges more slowly, but always converges to the right answer[13].

[12] When the model is wrong, the optimal parameter value θ_0 is often called the **pseudo-truth**.

[13] It is natural to wonder if one couldn't do better than kernel smoothing's $O(n^{-4/5})$ while still having no asymptotic bias. Resolving this is very difficult, but the answer turns out to be "no" in the

This doesn't change much if we use cross-validation. Writing $\widehat{h_{CV}}$ for the bandwidth picked by cross-validation, it turns out (Simonoff, 1996, ch. 5) that

$$\frac{\widehat{h_{CV}} - h_{\text{opt}}}{h_{\text{opt}}} - 1 = O(n^{-1/10}) \tag{4.21}$$

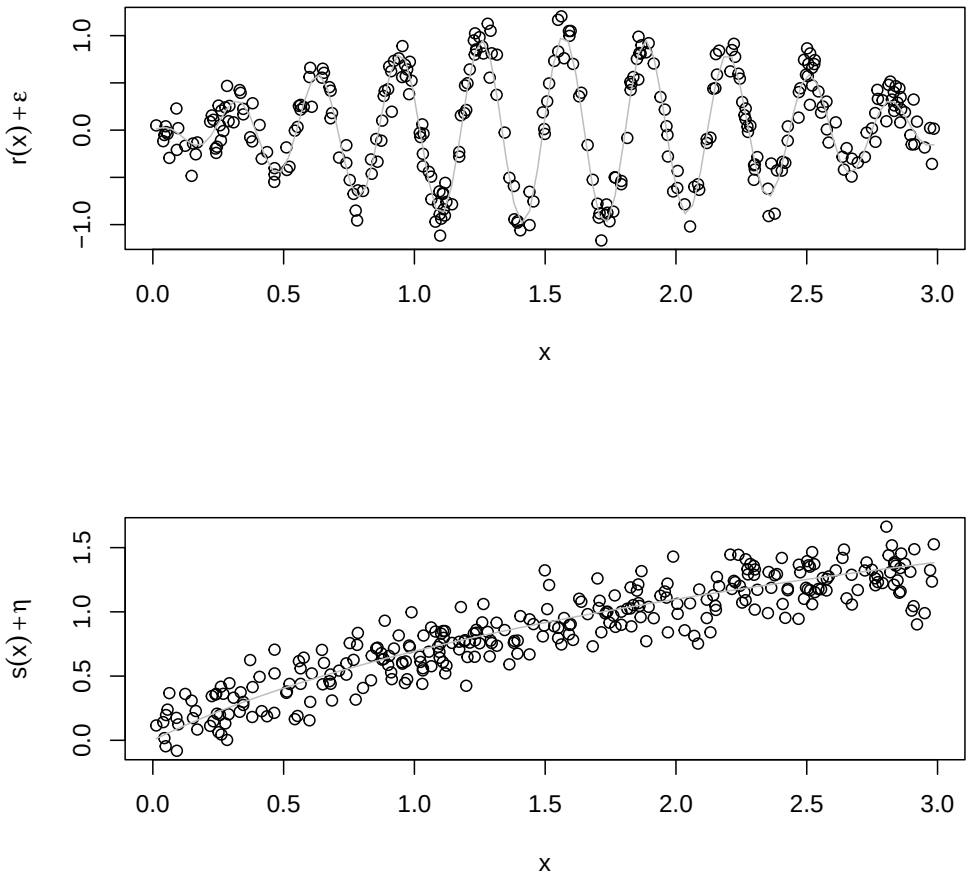
Given this, one concludes (Exercise 4.2) that the MSE of using $\widehat{h_{CV}}$ is also $O(n^{-4/5})$.

4.2.3 Summary on Kernel Smoothing in 1D

Suppose that X and Y are both one-dimensional, and the true regression function $\mu(x) = \mathbb{E}[Y|X = x]$ is continuous and has first and second derivatives[14]. Suppose that the noise around the true regression function is uncorrelated between different observations. Then the bias of kernel smoothing, when the kernel has bandwidth h, is $O(h^2)$, and the variance, after n samples, is $O((1/nh)^{-1})$. The optimal bandwidth is $O(n^{-1/5})$, and the excess mean squared error of using this bandwidth is $O(n^{-4/5})$. If the bandwidth is selected by cross-validation, the excess risk is still $O(n^{-4/5})$.

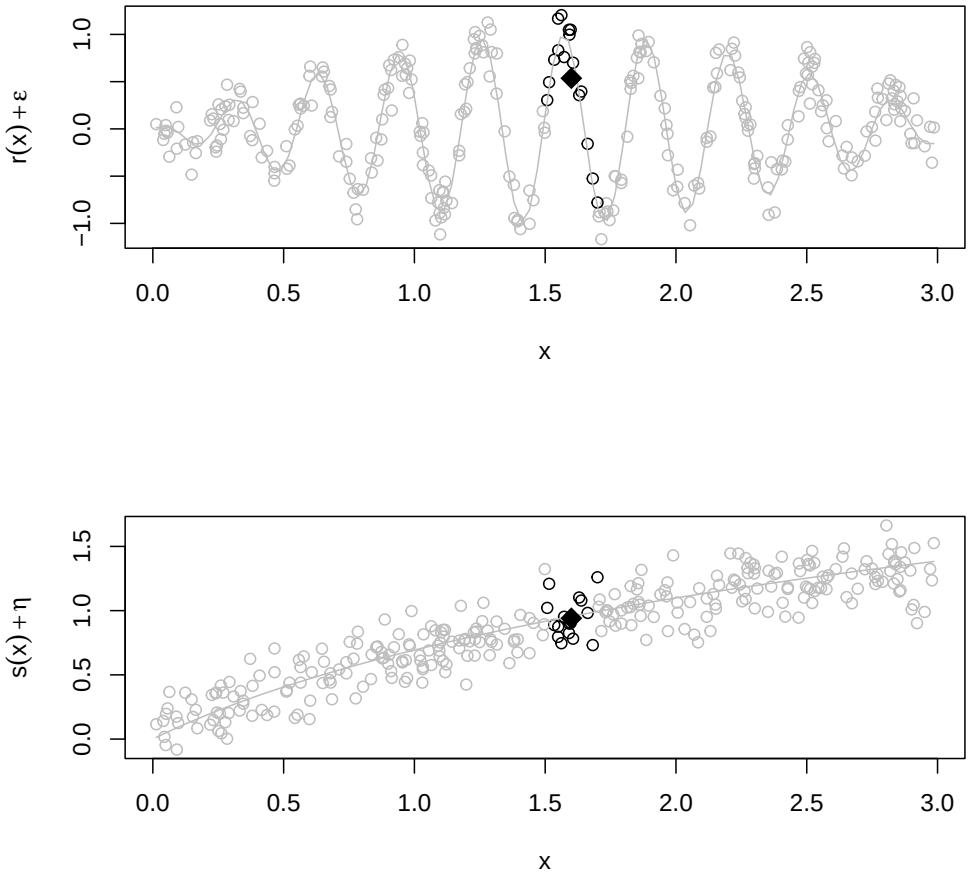
following sense (Wasserman, 2006). Any curve-fitting method which can learn arbitrary smooth regression functions will have some curves where it cannot converge any faster than $O(n^{-4/5})$. (In the jargon, that is the **minimax rate**.) Methods which converge faster than this for some kinds of curves have to converge more slowly for others. So this is the best rate we can hope for on truly unknown curves.

[14] Or can be approximated arbitrarily closely by such functions.




```
x = runif(300, 0, 3)
yr = true.r(x) + rnorm(length(x), 0, 0.15)
ys = true.s(x) + rnorm(length(x), 0, 0.15)
par(mfcol = c(2, 1))
plot(x, yr, xlab = "x", ylab = expression(r(x) + epsilon))
curve(true.r(x), col = "grey", add = TRUE)
plot(x, ys, xlab = "x", ylab = expression(s(x) + eta))
curve(true.s(x), col = "grey", add = TRUE)
```

Figure 4.4 The curves of Fig. 4.3 (in grey), plus IID Gaussian noise with mean 0 and standard deviation 0.15. The two curves are sampled at the same x values, but with different noise realizations.




```
par(mfcol = c(2, 1))
x.focus <- 1.6
x.lo <- x.focus - 0.1
x.hi <- x.focus + 0.1
colors = ifelse((x < x.hi) & (x > x.lo), "black", "grey")
plot(x, yr, xlab = "x", ylab = expression(r(x) + epsilon), col = colors)
curve(true.r(x), col = "grey", add = TRUE)
points(x.focus, mean(yr[(x < x.hi) & (x > x.lo)]), pch = 18, cex = 2)
plot(x, ys, xlab = "x", ylab = expression(s(x) + eta), col = colors)
curve(true.s(x), col = "grey", add = TRUE)
points(x.focus, mean(ys[(x < x.hi) & (x > x.lo)]), pch = 18, cex = 2)
par(mfcol = c(1, 1))
```

Figure 4.5 Relationship between smoothing and function roughness. In both panels we estimate the value of the regression function at $x = 1.6$ by averaging observations where $1.5 < x_i < 1.7$ (black points, others are "ghosted" in grey). The location of the average in shown by the large black diamond. This works poorly for the rough function r in the upper panel (the bias is large), but much better for the smoother function in the lower panel (the bias is small).

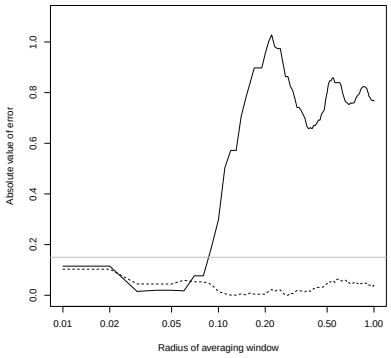
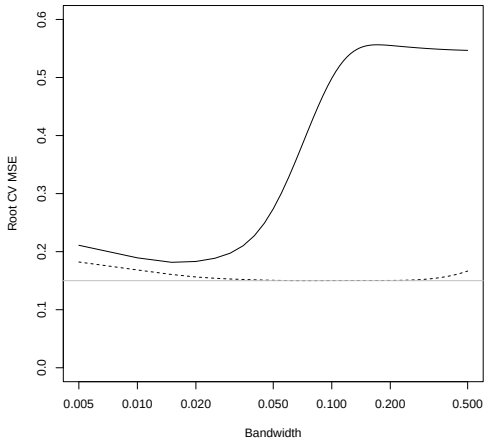
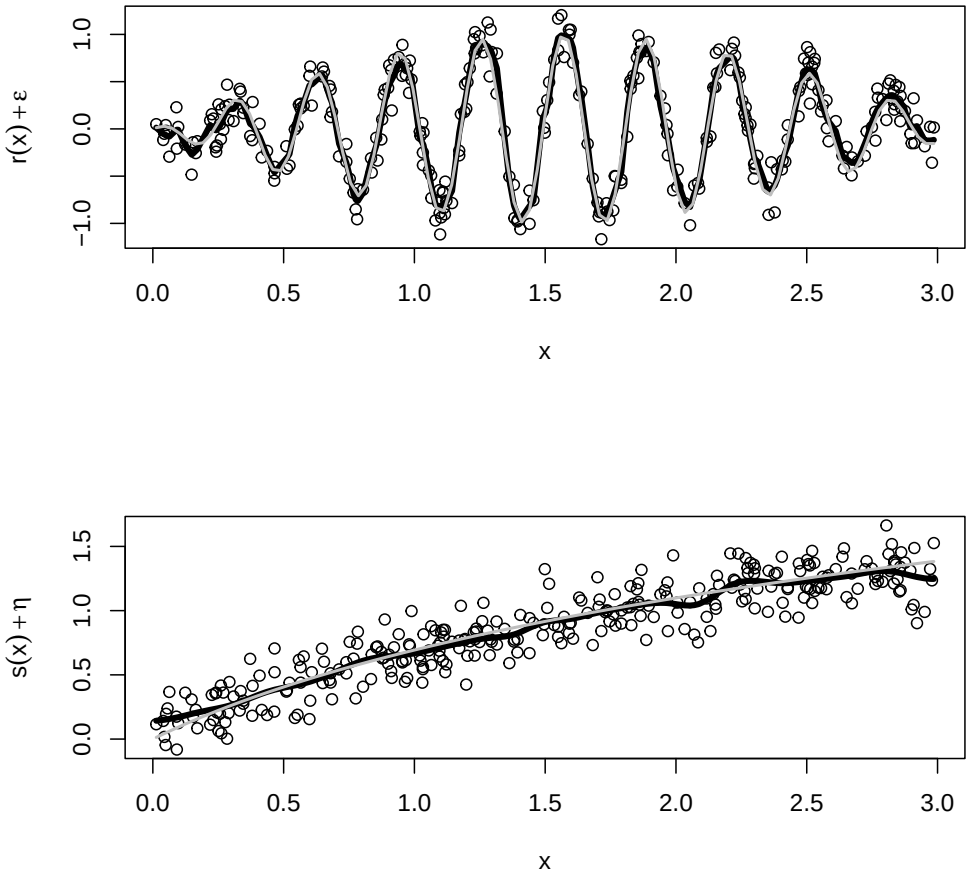


Figure 4.6 Error of estimating $r(1.6)$ (solid line) and $s(1.6)$ (dashed) from averaging observed values at $1.6 - h < x < 1.6 + h$, for different radii h. The grey is σ, the standard deviation of the noise — how can the estimation error be smaller than that?




```
rbws <- cv_bws_npreg(x, yr, bandwidths = (1:100)/200)
sbws <- cv_bws_npreg(x, ys, bandwidths = (1:100)/200)
plot(1:100/200, sqrt(rbws$CV_MSEs), xlab = "Bandwidth", ylab = "Root CV MSE", type = "l",
    ylim = c(0, 0.6), log = "x")
lines(1:100/200, sqrt(sbws$CV_MSEs), lty = "dashed")
abline(h = 0.15, col = "grey")
```

Figure 4.7 Cross-validated estimate of the (root) mean-squard error as a function of the bandwidth (solid curve, r data; dashed, s data; grey line, true noise σ). Notice that the rougher curve is more sensitive to the choice of bandwidth, and that the smoother curve is more predictable at every choice of bandwidth. CV selects bandwidths of 0.015 for r and 0.075 for s.




```
x.ord = order(x)
par(mfcol = c(2, 1))
plot(x, yr, xlab = "x", ylab = expression(r(x) + epsilon))
rhat <- npreg(bws = rbws$best.bw, txdat = x, tydat = yr)
lines(x[x.ord], fitted(rhat)[x.ord], lwd = 4)
curve(true.r(x), col = "grey", add = TRUE, lwd = 2)
plot(x, ys, xlab = "x", ylab = expression(s(x) + eta))
shat <- npreg(bws = sbws$best.bw, txdat = x, tydat = ys)
lines(x[x.ord], fitted(shat)[x.ord], lwd = 4)
curve(true.s(x), col = "grey", add = TRUE, lwd = 2)
par(mfcol = c(1, 1))
```

Figure 4.8 Data from the running examples (circles), true regression functions (grey) and kernel estimates of regression functions with CV-selected bandwidths (black). R NOTES: The `x` values aren't sorted, so we need to put them in order before drawing lines connecting the fitted values; then we need to put the fitted values in the same order. Alternately, we could have used `predict` on the sorted values, as in §4.3.

4.3 Kernel Regression with Multiple Inputs

For the most part, when I've been writing out kernel regression I have been treating the input variable x as a scalar. There's no reason to insist on this, however; it could equally well be a vector. If we want to enforce that in the notation, say by writing $\vec{x} = (x^1, x^2, \ldots x^d)$, then the kernel regression of y on $\vec{x}$ would just be

$$\widehat{\mu}(\vec{x}) = \sum_{i=1}^{n} y_i \frac{K(\vec{x} - \vec{x_i})}{\sum_{j=1}^{n} K(\vec{x} - \vec{x_j})} \tag{4.22}$$

In fact, if we want to predict a vector, we'd just substitute $\vec{y_i}$ for y_i above.

To make this work, we need kernel functions for vectors. For scalars, I said that any probability density function would work so long as it had mean zero, and a finite, strictly positive (not 0 or ∞) variance. The same conditions carry over: any distribution over vectors can be used as a multivariate kernel, provided it has mean zero, and the variance matrix is finite and "positive definite"[15]. In practice, the overwhelmingly most common and practical choice is to use **product kernels**[16].

A product kernel simply uses a different kernel for each component, and then multiplies them together:

$$K(\vec{x} - \vec{x_i}) = K_1(x^1 - x_i^1) K_2(x^2 - x_i^2) \ldots K_d(x^d - x_i^d) \tag{4.23}$$

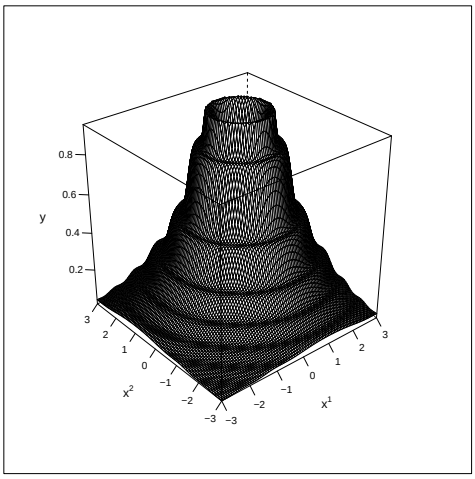
Now we just need to pick a bandwidth for each kernel, which in general should not be equal — say $\vec{h} = (h_1, h_2, \ldots h_d)$. Instead of having a one-dimensional error curve, as in Figure 4.1 or 4.2, we will have a d-dimensional error surface, but we can still use cross-validation to find the vector of bandwidths that generalizes best. We generally can't, unfortunately, break the problem up into somehow picking the best bandwidth for each variable without considering the others. This makes it slower to select good bandwidths in multivariate problems, but still often feasible.

(We can actually turn the need to select bandwidths together to our advantage. If one or more of the variables are irrelevant to our prediction given the others, cross-validation will tend to give them the maximum possible bandwidth, and smooth away their influence. In Chapter 14, we'll look at formal tests based on this idea.)

Kernel regression will recover almost any regression function. This is true even when the true regression function involves lots of interactions among the input variables, perhaps in complicated forms that would be very hard to express in linear regression. For instance, Figure 4.9 shows a contour plot of a reasonably complicated regression surface, at least if one were to write it as polynomials in x^1 and x^2, which would be the usual approach. Figure 4.11 shows the estimate we get with a product of Gaussian kernels and only 1000 noisy data points. It's

[15] Remember that for a matrix $\mathbf{v}$ to be "positive definite", it must be the case that for any vector $\vec{a} \neq \vec{0}$, $\vec{a} \cdot \mathbf{v}\vec{a} > 0$. Covariance matrices are automatically non-negative, so we're just ruling out the case of some weird direction along which the distribution has zero variance.

[16] People do sometimes use multivariate Gaussians with non-trivial correlation across the variables, but this is very rare in my experience.

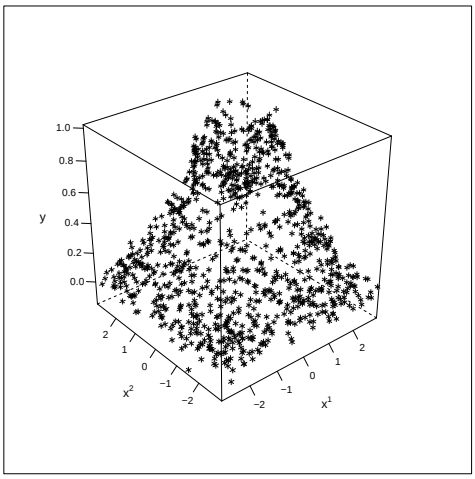



```
x1.points <- seq(-3, 3, length.out = 100)
x2.points <- x1.points
x12grid <- expand.grid(x1 = x1.points, x2 = x2.points)
y <- matrix(0, nrow = 100, ncol = 100)
y <- outer(x1.points, x2.points, f)
library(lattice)
wireframe(y ~ x12grid$x1 * x12grid$x2, scales = list(arrows = FALSE), xlab = expression(x^1),
    ylab = expression(x^2), zlab = "y")
```

Figure 4.9 An example of a regression surface that would be very hard to learn by piling together interaction terms in a linear regression framework. (Can you guess what the mystery function `f` is?) — `wireframe` is from the graphics library `lattice`.

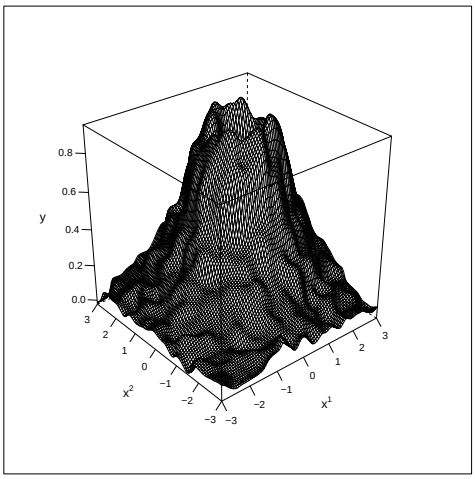
not perfect, of course (in particular the estimated contours aren't as perfectly smooth and round as the true ones), but the important thing is that we got this without having to know, and describe in Cartesian coordinates, the type of shape we were looking for. Kernel smoothing *discovered* the right general form.

There are limits to these abilities of kernel smoothers; the biggest one is that they require more and more data as the number of predictor variables increases. We will see later (Chapter 8) exactly how much data is required, generalizing the kind of analysis done §4.2.2, and some of the compromises this can force us into.




```
x1.noise <- runif(1000, min = -3, max = 3)
x2.noise <- runif(1000, min = -3, max = 3)
y.noise <- f(x1.noise, x2.noise) + rnorm(1000, 0, 0.05)
noise <- data.frame(y = y.noise, x1 = x1.noise, x2 = x2.noise)
cloud(y ~ x1 * x2, data = noise, col = "black", scales = list(arrows = FALSE), xlab = expression(x^1
    ylab = expression(x^2), zlab = "y")
```

Figure 4.10 1000 points sampled from the surface in Figure 4.9, plus independent Gaussian noise (s.d. = 0.05).




```
noise.np <- npreg(y ~ x1 + x2, data = noise)
y.out <- matrix(0, 100, 100)
y.out <- predict(noise.np, newdata = x12grid)
wireframe(y.out ~ x12grid$x1 * x12grid$x2, scales = list(arrows = FALSE), xlab = expression(x^1),
    ylab = expression(x^2), zlab = "y")
```

Figure 4.11 Gaussian kernel regression of the points in Figure 4.10. Notice that the estimated function will make predictions at arbitrary points, not just the places where there was training data.

4.4 Interpreting Smoothers: Plots

In a linear regression without interactions, it is fairly easy to interpret the coefficients. The expected response changes by β_i for a one-unit change in the ith input variable. The coefficients are also the derivatives of the expected response with respect to the inputs. And it is easy to draw pictures of how the output changes as the inputs are varied, though the pictures are somewhat boring (straight lines or planes).

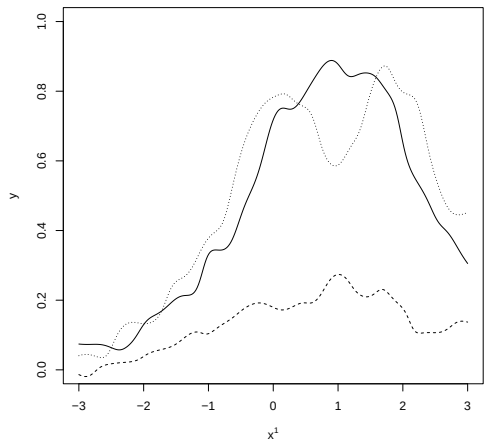
As soon as we introduce interactions, all this becomes harder, even for parametric regression. If there is an interaction between two components of the input, say x^1 and x^2, then we can't talk about the change in the expected response for a one-unit change in x^1 without saying what x^2 is. We might *average* over x^2 values, and in §4.5 below we'll see next time a reasonable way of doing this, but the flat statement "increasing x^1 by one unit increases the response by β_1" is just false, no matter what number we fill in for β_1. Likewise for derivatives; we'll come back to them next time as well.

What about pictures? With only two input variables, we can make wireframe plots like Figure 4.11, or contour or level plots, which will show the predictions for different combinations of the two variables. But what if we want to look at one variable at a time, or there are more than two input variables?

A reasonable way to produce *a* curve for each input variable is to set all the others to some "typical" value, like their means or medians, and to then plot the predicted response as a function of the one remaining variable of interest (Figure 4.12). Of course, when there are interactions, changing the values of the other inputs will change the response to the input of interest, so it's a good idea to produce a couple of curves, possibly super-imposed (Figure 4.12 again).

If there are three or more input variables, we can look at the interactions of any two of them, taken together, by fixing the others and making three-dimensional or contour plots, along the same principles.

The fact that smoothers don't give us a simple story about how each input is associated with the response may seem like a disadvantage compared to using linear regression. Whether it really is a disadvantage depends on whether there really is a simple story to be told, and/or how much big a lie you are prepared to tell in order to keep your story simple.




```
new.frame <- data.frame(x1 = seq(-3, 3, length.out = 300), x2 = median(x2.noise))
plot(new.frame$x1, predict(noise.np, newdata = new.frame), type = "l", xlab = expression(x^1),
    ylab = "y", ylim = c(0, 1))
new.frame$x2 <- quantile(x2.noise, 0.25)
lines(new.frame$x1, predict(noise.np, newdata = new.frame), lty = 2)
new.frame$x2 <- quantile(x2.noise, 0.75)
lines(new.frame$x1, predict(noise.np, newdata = new.frame), lty = 3)
```

Figure 4.12 Predicted mean response as function of the first input coordinate x^1 for the example data, evaluated with the second coordinate x^2 set to the median (solid), its 25th percentile (dashed) and its 75th percentile (dotted). Note that the changing shape of the partial response curve indicates an interaction between the two inputs. Also, note that the model can make predictions at arbitrary coordinates, whether or not there were any training points there.

4.5 Average Predictive Comparisons

Suppose we have a linear regression model

$$Y = \beta_1 X_1 + \beta_2 X_2 + \epsilon \tag{4.24}$$

and we want to know how much Y changes, on average, for a one-unit increase in X_1. The answer, as you know very well, is just β_1:

$$[\beta_1(X_1 + 1) + \beta_2 X_2] - [\beta_1 X_1 + \beta_2 X_2] = \beta_1 \tag{4.25}$$

This is an interpretation of the regression coefficients which you are very used to giving. But it fails as soon as we have interactions:

$$Y = \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_1 X_2 + \epsilon \tag{4.26}$$

Now the effect of increasing X_1 by 1 is

$$[\beta_1(X_1+1)+\beta_2 X_2+\beta_3(X_1+1)X_2]-[\beta_1 X_1+\beta_2 X_2+\beta_3 X_1 X_2] = \beta_1+\beta_3 X_2 \tag{4.27}$$

The right answer to "how much does the response change when X_1 is increased by one unit?" depends on the value of X_2; it's certainly not just "β_1".

We also can't give just a single answer if there are nonlinearities. Suppose that the true regression function is this:

$$Y = \frac{e^{\beta X}}{1 + e^{\beta X}} + \epsilon \tag{4.28}$$

which looks like Figure 4.13, setting $\beta = 7$ (for luck). Moving x from -4 to -3 increases the response by 7.57×10^{-10}, but the increase in the response from $x = -1$ to $x = 0$ is 0.499. Functions like this are very common in psychology, medicine (dose-response curves for drugs), biology, etc., and yet we cannot sensibly talk about *the* response to a one-unit increase in x. (We will come back to curves which look like this in Chapter 11.)

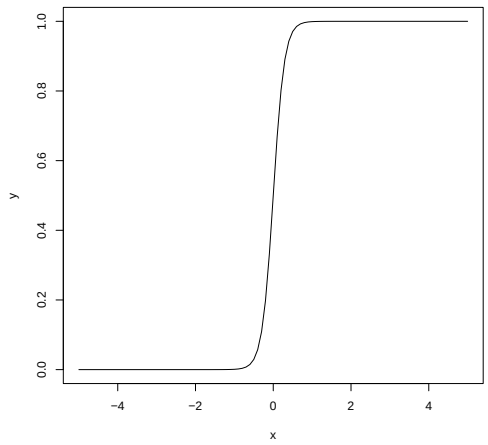
More generally, let's say we are regressing Y on a vector $\vec{X}$, and want to assess the impact of one component of the input on Y. To keep the use of subscripts and superscripts to a minimum, we'll write $\vec{X} = (U, \vec{V})$, where U is the coordinate we're really interested in. (It doesn't have to come first, of course.) We would like to know how much the prediction changes as we change u,

$$\mathbb{E}\left[Y|\vec{X} = (u^{(2)}, \vec{v})\right] - \mathbb{E}\left[Y|\vec{X} = (u^{(1)}, \vec{v})\right] \tag{4.29}$$

and the change in the response per unit change in u,

$$\frac{\mathbb{E}\left[Y|\vec{X} = (u^{(2)}, \vec{v})\right] - \mathbb{E}\left[Y|\vec{X} = (u^{(1)}, \vec{v})\right]}{u^{(2)} - u^{(1)}} \tag{4.30}$$

Both of these, but especially the latter, are called the **predictive comparison**. Note that both of them, as written, depend on $u^{(1)}$ (the starting value for the variable of interest), on $u^{(2)}$ (the ending value), and on $\vec{v}$ (the other variables, held fixed during this comparison). We have just seen that in a linear model




```
curve(exp(7 * x)/(1 + exp(7 * x)), from = -5, to = 5, ylab = "y")
```

Figure 4.13 The function of Eq. 4.28, with $\beta = 7$.

without interactions, $u^{(1)}$, $u^{(2)}$ and $\vec{v}$ all go away and leave us with the regression coefficient on u. In nonlinear or interacting models, we can't simplify so much.

Once we have estimated a regression model, we can choose our starting point, ending point and context, and just plug in to Eq. 4.29 or Eq. 4.30. (Or problem 9 in problem set 11.) But suppose we do want to boil this down into a single number for each input variable — how might we go about this?

One good answer, which comes from Gelman and Pardoe (2007), is just to average 4.30 over the data[17]. More specifically, we have as our **average predictive comparison** for u

$$\frac{\sum_{i=1}^{n}\sum_{j=1}^{n}(\widehat{\mu}(u_j,\vec{v}_i)-\widehat{\mu}(u_i,\vec{v}_i))\text{sign}(u_j-u_i)}{\sum_{i=1}^{n}\sum_{j=1}^{n}(u_j-u_i)\text{sign}(u_j-u_i)} \tag{4.31}$$

where i and j run over data points, $\widehat{\mu}$ is our estimated regression function, and the sign function is defined by $\text{sign}(x) = +1$ if $x > 0$, $= 0$ if $x = 0$, and $= -1$ if $x < 0$. We use the sign function this way to make sure we are always looking at the consequences of *increasing* u.

The average predictive comparison is a reasonable summary of how rapidly we should expect the response to vary as u changes slightly. But we need to remember that once the model is nonlinear or has interactions, it's just not possible to boil down the whole predictive relationship between u and y into one number. In particular, the value of Eq. 4.31 is going to depend on the distribution of u (and possibly of v), even when the regression function is unchanged. (See Exercise 4.3.)

[17] Actually, they propose something a bit more complicated, which takes into account the uncertainty in our estimate of the regression function, via bootstrapping (Chapter 6).

```
make.demo.df <- function(n) {
    demo.func <- function(x, z, w) {
        20 * x^2 + ifelse(w == "A", z, 10 * exp(z)/(1 + exp(z)))
    }
    x <- runif(n, -1, 1)
    z <- rnorm(n, 0, 10)
    w <- sample(c("A", "B"), size = n, replace = TRUE)
    y <- demo.func(x, z, w) + rnorm(n, 0, 0.05)
    return(data.frame(x = x, y = y, z = z, w = w))
}

demo.df <- make.demo.df(100)
```

Code Example 5: *Generating data from Eq. 4.32.*

4.6 Computational Advice: `npreg`

The homework will call for you to do nonparametric regression with the `np` package — which we've already looked at a little. It's a powerful bit of software, but it can take a bit of getting used to. This section is not a substitute for reading Hayfield and Racine (2008), but should get you started.

We'll look at a synthetic-data example with four variables: a quantitative response Y, two quantitative predictors X and Z, and a categorical predictor W, which can be either "A" or "B". The true model is

$$Y = \epsilon + 20X^2 + \begin{cases} Z & \text{if } W = A \\ 10e^Z/(1+e^Z) & \text{if } W = B \end{cases} \tag{4.32}$$

with $\epsilon \sim \mathcal{N}(0, 0.05)$. Code Example 5 generates some data from this model for us.

The basic function for fitting a kernel regression in `np` is `npreg` — conceptually, it's the equivalent of `lm`. Like `lm`, it takes a `formula` argument, which specifies the model, and a `data` argument, which is a data frame containing the variables included in the formula. The basic idea is to do something like this:

```
demo.np1 <- npreg(y ~ x + z, data = demo.df)
```

The variables on the right-hand side of the formula are the predictors; we use `+` to separate them. Kernel regression will automatically include interactions between all variables, so there is no special notation for interactions. Similarly, there is no point in either including or excluding intercepts. If we wanted to transform either a predictor variable or the response, as in `lm`, we can do so. Run like this, `npreg` will try to determine the best bandwidths for the predictor variables, based on a sophisticated combination of cross-validation and optimization.

Let's look at the output of `npreg`:

```
summary(demo.np1)
##
## Regression Data: 100 training points, in 2 variable(s)
##                         x        z
## Bandwidth(s): 0.06227118 4.744557
##
```

```
## Kernel Regression Estimator: Local-Constant
## Bandwidth Type: Fixed
## Residual standard error: 2.584642
## R-squared: 0.9378975
##
## Continuous Kernel Type: Second-Order Gaussian
## No. Continuous Explanatory Vars.: 2
```

The main things here are the bandwidths. We also see the root mean squared error on the training data. Note that this is the in-sample root MSE; if we wanted the in-sample MSE, we could do

```
demo.np1$MSE
## [1] 6.680373
```

(You can check that this is the square of the residual standard error above.) If we want the cross-validated MSE used to pick the bandwidths, that's

```
demo.np1$bws$fval
## [1] 25.52361
```

The `fitted` and `residuals` functions work on these objects just like they do in `lm` objects, while the `coefficients` and `confint` functions do not. (Why?)

The `predict` function also works like it does for `lm`, expecting a data frame containing columns whose names match those in the formula used to fit the model:

```
predict(demo.np1, newdata = data.frame(x = -1, z = 5))
## [1] 26.04758
```

With two predictor variables, there is a nice three-dimensional default plot (Figure 4.14).

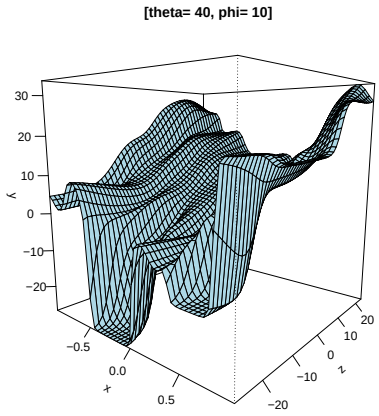
Kernel functions can also be defined for categorical and ordered variables. These can be included in the formula by wrapping the variable in `factor()` or `ordered()`, respectively:

```
demo.np3 <- npreg(y ~ x + z + factor(w), data = demo.df)
```

Again, there's no point, or need, to indicate interactions. Including the extra variable, not surprisingly, improves the cross-validated MSE:

```
demo.np3$bws$fval
## [1] 13.94945
```

With three or more predictor variables, we'd need a four-dimensional plot, which is hard. Instead, the default is to plot what happens as we sweep one variable with the others held fixed (by default, at their medians; see `help(npplot)` for changing that), as in Figure 4.15. We get something parabola-ish as we sweep X (which is right), and something near a step function as we sweep Z (which is right when $W = B$), so we're not doing badly for estimating a fairly complicated function of three variables with only 100 samples. We could also try fixing W at one value or another and making a perspective plot — Figure 4.16.




```
plot(demo.np1, theta = 40, view = "fixed")
```

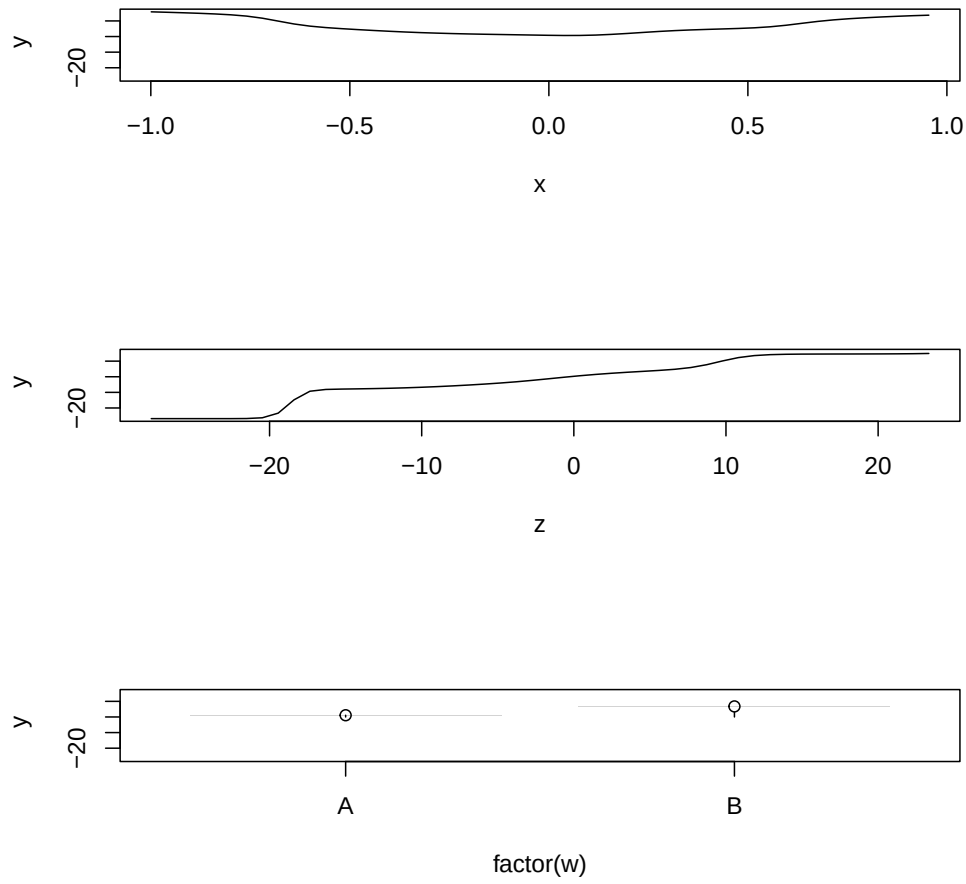
Figure 4.14 Plot of the kernel regression with just two predictor variables. (See `help(npplot)` for plotting options.

The default optimization of bandwidths is *extremely* aggressive. It keeps adjusting the bandwidths until the changes in the cross-validated MSE are very small, or the changes in the bandwidths themselves are very small. The "tolerances" for what count as "very small" are controlled by arguments to `npreg` called `tol` (for the bandwidths) and `ftol` (for the MSE), which default to about 10^{-8} and 10^{-7}, respectively. With a lot of data, or a lot of variables, this gets *extremely* slow. One can often make `npreg` run much faster, with no real loss of accuracy, by adjusting these options. A decent rule of thumb is to start with `tol` and `ftol` both at 0.01. One can use the bandwidth found by this initial coarse search to start a more refined one, as follows:

```
bigdemo.df <- make.demo.df(1000)
system.time(demo.np4 <- npreg(y ~ x + z + factor(w), data = bigdemo.df, tol = 0.01,
    ftol = 0.01))
##    user  system elapsed
##  31.314   0.330  32.160
```

This tells us how much time it took R to run `npreg`, dividing that between time spent exclusively on our job and on background system tasks. The result of the run is stored in `demo.np4`:

```
demo.np4$bws
##
## Regression Data (1000 observations, 3 variable(s)):
##
##                       x        z    factor(w)
```




```
plot(demo.np3)
```

Figure 4.15 Predictions of `demo.np3` as each variable is swept over its range, with the others held at their medians.

```
## Bandwidth(s): 0.06101409 2.441987 9.649056e-08
##
## Regression Type: Local-Constant
## Bandwidth Selection Method: Least Squares Cross-Validation
## Formula: y ~ x + z + factor(w)
## Bandwidth Type: Fixed
## Objective Function Value: 1.517143 (achieved on multistart 1)
##
## Continuous Kernel Type: Second-Order Gaussian
```

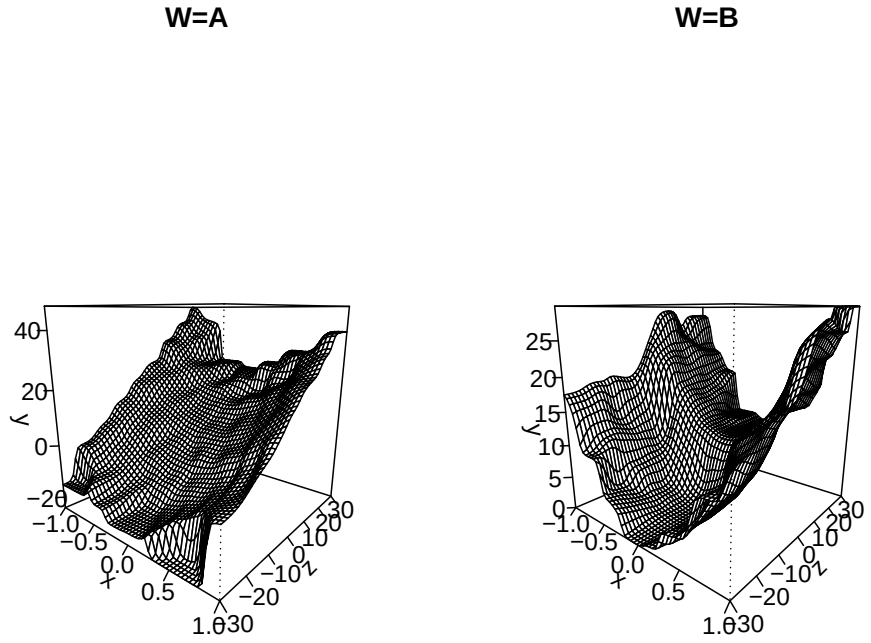
```
## No. Continuous Explanatory Vars.: 2
##
## Unordered Categorical Kernel Type: Aitchison and Aitken
## No. Unordered Categorical Explanatory Vars.: 1
```

The bandwidths have all shrunk (as they should), and the cross-validated MSE is also much smaller (1.5 versus 14 before). Figure 4.16 shows the estimated regression surfaces for both values of the categorical variable.

The package also contains a function, `npregbw`, which takes a formula and a data frame, and just optimizes the bandwidth. This is called automatically by `npreg`, and many of the relevant options are documented in its help page. One can also use the output of `npregbw` as an argument to `npreg`, in place of a formula.

As a final piece of computational advice, you will notice when you run these commands yourself that the bandwidth-selection functions by default print out lots of progress-report messages. This can be annoying, especially if you are embedding the computation in a document, and so can be suppressed by setting a global option at the start of your code:

```
options(np.messages = FALSE)
```




```
x.seq <- seq(from = -1, to = 1, length.out = 50)
z.seq <- seq(from = -30, to = 30, length.out = 50)
grid.A <- expand.grid(x = x.seq, z = z.seq, w = "A")
grid.B <- expand.grid(x = x.seq, z = z.seq, w = "B")
yhat.A <- predict(demo.np4, newdata = grid.A)
yhat.B <- predict(demo.np4, newdata = grid.B)
par(mfrow = c(1, 2))
persp(x = x.seq, y = z.seq, z = matrix(yhat.A, nrow = 50), theta = 40, main = "W=A",
    xlab = "x", ylab = "z", zlab = "y", ticktype = "detailed")
persp(x = x.seq, y = z.seq, z = matrix(yhat.B, nrow = 50), theta = 40, main = "W=B",
    xlab = "x", ylab = "z", zlab = "y", ticktype = "detailed")
```

Figure 4.16 The regression surfaces learned for the demo function at the two different values of the categorical variable. Note that holding z fixed, we always see a parabolic shape as we move along x (as we should), while whether we see a line or something close to a step function at constant x depends on w, as it should.

4.7 Further Reading

Simonoff (1996) is a good practical introduction to kernel smoothing and related methods. Wasserman (2006) provides more theory. Li and Racine (2007) is a detailed treatment of nonparametric methods for econometric problems, overwhelmingly focused on kernel regression and kernel density estimation (which we'll get to in Chapter 14); Racine (2008) summarizes.

While kernels are a nice, natural method of non-parametric smoothing, they are not the only one. We saw nearest-neighbors in §1.5.1, and will encounter splines (continuous piecewise-polynomial models) in Chapter 7 and trees (piecewise-constant functions, with cleverly chosen pieces) in Chapter 13; local linear models (§10.5) combine kernels and linear models. There are many, many more options.

Historical Notes

Kernel regression was introduced, independently, by Nadaraya (1964) and Watson (1964); both were inspired by kernel density estimation.

In the mid-2010s, kernel smoothing was re-invented by computer scientists working on large language models, under the curious name of "attention". This turned out to be a key technical step in creating language models like GPT (Vaswani *et al.*, 2017). The first people to realize that "attention", in this sense, was a kind of kernel smoothing seem to have been Tsai *et al.* (2019).

Exercises

4.1 Suppose we use a uniform ("boxcar") kernel extending over the region $(-h/2, h/2)$. Show that

$$\mathbb{E}\left[\widehat{\mu}(0)\right] = \mathbb{E}\left[\mu(X) \middle| X \in \left(-\frac{h}{2}, \frac{h}{2}\right)\right] \tag{4.33}$$

$$= \mu(0) + \mu'(0)\mathbb{E}\left[X \middle| X \in \left(-\frac{h}{2}, \frac{h}{2}\right)\right] + \frac{\mu''(0)}{2}\mathbb{E}\left[X^2 \middle| X \in \left(-\frac{h}{2}, \frac{h}{2}\right)\right] + o(h^2) \tag{4.34}$$

Show that $\mathbb{E}\left[X \middle| X \in \left(-\frac{h}{2}, \frac{h}{2}\right)\right] = O(f'(0)h^2)$, and that $\mathbb{E}\left[X^2 \middle| X \in \left(-\frac{h}{2}, \frac{h}{2}\right)\right] = O(h^2)$. Conclude that the over-all bias is $O(h^2)$.

4.2 Use Eqs. 4.21, 4.17 and 4.16 to show that the excess risk of the kernel smoothing, when the bandwidth is selected by cross-validation, is also $O(n^{-4/5})$.

4.3 Generate 1000 data points where X is uniformly distributed between -4 and 4, and $Y = e^{7x}/(1+e^{7x}) + \epsilon$, with ϵ Gaussian and with variance 0.01. Use non-parametric regression to estimate $\widehat{\mu}(x)$, and then use Eq. 4.31 to find the average predictive comparison. Now re-run the simulation with X uniform on the interval $[0, 0.5]$ and re-calculate the average predictive comparison. What happened?

5

Simulation

You will recall from your previous statistics courses that quantifying uncertainty in statistical inference requires us to get at the **sampling distributions** of things like estimators. When the very strong simplifying assumptions of basic statistics courses do not apply[1], there is little hope of being able to write down sampling distributions in closed form. There is equally little help when the estimates are themselves complex objects, like kernel regression curves or even histograms, rather than short, fixed-length parameter vectors. We get around this by using simulation to approximate the sampling distributions we can't calculate.

5.1 What Is a Simulation?

A mathematical model is a mathematical story about how the data could have been made, or **generated**. Simulating the model means following that story, implementing it, step by step, in order to produce something which should look like the data — what's sometimes called **synthetic data**, or **surrogate data**, or a **realization** of the model. In a stochastic model, some of the steps we need to follow involve a random component, and so multiple simulations starting from exactly the same inputs or initial conditions will not give exactly the same outputs or realizations. Rather, the model specifies a distribution over the realizations, and doing many simulations gives us a good approximation to this distribution.

For a trivial example, consider a model with three random variables, $X_1 \sim \mathcal{N}(\mu_1, \sigma_1^2)$, $X_2 \sim \mathcal{N}(\mu_2, \sigma_2^2)$, with $X_1 \perp\!\!\!\perp X_2$, and $X_3 = X_1 + X_2$. Simulating from this model means drawing a random value from the first normal distribution for X_1, drawing a second random value for X_2, and adding them together to get X_3. The marginal distribution of X_3, and the joint distribution of (X_1, X_2, X_3), are implicit in this specification of the model, and we can find them by running the simulation.

In this particular case, we could also find the distribution of X_3, and the joint distribution, by probability calculations of the kind you learned how to do in your basic probability courses. For instance, X_3 is $\mathcal{N}(\mu_1 + \mu_2, \sigma_1^2 + \sigma_2^2)$. These

[1] As discussed *ad nauseam* in Chapter 2, in your linear models class, you learned about the sampling distribution of regression coefficients when the linear model is true, and the noise is Gaussian, independent of the predictor variables, and has constant variance. As an exercise, try to get parallel results when the noise has a t distribution with 10 degrees of freedom.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

analytical probability calculations can usually be thought of as just short-cuts for exhaustive simulations.

5.2 How Do We Simulate Stochastic Models?

5.2.1 Chaining Together Random Variables

Stochastic models are usually specified by sets of conditional distributions for one random variable, given some other variable or variables. For instance, a simple linear regression model might have the specification

$$X \sim \mathcal{U}(x_{\min}, x_{\max}) \tag{5.1}$$

$$Y|X \sim \mathcal{N}(\beta_0 + \beta_1 X, \sigma^2) \tag{5.2}$$

If we knew how to generate a random variable from the distributions given on the right-hand sides, we could simulate the whole model by chaining together draws from those conditional distributions. This is in fact the general strategy for simulating any sort of stochastic model, by chaining together random variables.[2]

You might ask why we don't start by generating a random Y, and then generate X by drawing from the $X|Y$ distribution. The basic answer is that you could, but it would generally be messier. (Just try to work out the conditional distribution $X|Y$.) More broadly, in Chapter 18, we'll see how to arrange the variables in complicated probability models in a natural order, so that we start with independent, "exogenous" variables, then first-generation variables which only need to be conditioned on the exogenous variables, then second-generation variables which are conditioned on first-generation ones, and so forth. This is also the natural order for simulation.

The upshot is that we can reduce the problem of simulating to that of generating random variables.

5.2.2 Random Variable Generation

5.2.2.1 Built-in Random Number Generators

R provides random number generators for most of the most common distributions. By convention, the names of these functions all begin with the letter "r", followed by the abbreviation of the functions, and the first argument is always the number of draws to make, followed by the parameters of the distribution. Some examples:

```
rnorm(n, mean = 0, sd = 1)
runif(n, min = 0, max = 1)
rexp(n, rate = 1)
rpois(n, lambda)
rbinom(n, size, prob)
```

[2] In this case, we could in principle first generate Y, and then draw from $Y|X$, but have fun finding those distributions. Especially have fun if, say, X has a t distribution with 10 degrees of freedom. (I keep coming back to that idea, because it's really a very small change from being Gaussian.)

A further convention is that these parameters can be **vectorized**. Rather than giving a single mean and standard deviation (say) for multiple draws from the Gaussian distribution, each draw can have its own:

```
rnorm(10, mean = 1:10, sd = 1/sqrt(1:10))
```

That instance is rather trivial, but the exact same principle would be at work here:

```
rnorm(nrow(x), mean = predict(regression.model, newdata = x), sd = predict(volatility.model,
    newdata = x))
```

where `regression.model` and `volatility.model` are previously-defined parts of the model which tell us about conditional expectations and conditional variances.

Of course, none of this explains how R actually draws from any of these distributions; it's all at the level of a black box, which is to say black magic. Because ignorance is evil, and, even worse, *unhelpful* when we need to go beyond the standard distributions, it's worth opening the black box just a bit. We'll look at using transformations between distributions, and, in particular, transforming uniform distributions into others (§5.2.2.3).

5.2.2.2 Transformations

If we can generate a random variable Z with some distribution, and $V = g(Z)$, then we can generate V. So one thing which gets a lot of attention is writing random variables as transformations of one another — ideally as transformations of easy-to-generate variables.

Example: from standard to customized Gaussians

Suppose we can generate random numbers from the standard Gaussian distribution $Z \sim \mathcal{N}(0,1)$. Then we can generate from $\mathcal{N}(\mu, \sigma^2)$ as $\sigma Z + \mu$. We can generate χ^2 random variables with 1 degree of freedom as Z^2. We can generate χ^2 random variables with d degrees of freedom by summing d independent copies of Z^2.

In particular, if we can generate random numbers uniformly distributed between 0 and 1, we can use this to generate anything which is a transformation of a uniform distribution. How far does that extend?

5.2.2.3 Quantile Method

Suppose that we know the **quantile function** Q_Z for the random variable Z we want, so that $Q_Z(0.5)$ is the median of X, $Q_Z(0.9)$ is the 90th percentile, and in general $Q_Z(p)$ is bigger than or equal to Z with probability p. Q_Z comes as a pair with the cumulative distribution function F_Z, since

$$Q_Z(F_Z(a)) = a, \; F_Z(Q_Z(p)) = p \tag{5.3}$$

In the **quantile method** (or **inverse distribution transform method**), we generate a uniform random number U and feed it as the argument to Q_Z. Now

$Q_Z(U)$ has the distribution function F_Z:

$$\begin{aligned} \Pr\left(Q_Z(U) \leq a\right) &= \Pr\left(F_Z(Q_Z(U)) \leq F_Z(a)\right) && (5.4)\\ &= \Pr\left(U \leq F_Z(a)\right) && (5.5)\\ &= F_Z(a) && (5.6) \end{aligned}$$

where the last line uses the fact that U is uniform on $[0,1]$, and the first line uses the fact that F_Z is a non-decreasing function, so $b \leq a$ is true if and only if $F_Z(b) \leq F_Z(a)$.

Example: Exponentials

The CDF of the exponential distribution with rate λ is $1 - e^{-\lambda z}$. The quantile function $Q(p)$ is thus $-\frac{\log(1-p)}{\lambda}$. (Notice that this is positive, because $1-p < 1$ and so $\log(1-p) < 0$, and that it has units of $1/\lambda$, which are the units of z, as it should.) Therefore, if U Unif$(0,1)$, then $-\frac{\log(1-U)}{\lambda} \sim \text{Exp}(\lambda)$. This is the method used by `rexp()`.

Example: Power laws

The **Pareto distribution** or **power-law distribution** is a two-parameter family, $f(z;\alpha,z_0) = \frac{\alpha-1}{z_0}\left(\frac{z}{z_0}\right)^{-\alpha}$ if $z \geq z_0$, with density 0 otherwise. Integration shows that the cumulative distribution function is $F(z;\alpha,z_0) = 1 - \left(\frac{z}{z_0}\right)^{-\alpha+1}$. The quantile function therefore is $Q(p;\alpha,z_0) = z_0(1-p)^{-\frac{1}{\alpha-1}}$. (Notice that this has the same units as z, as it should.)

Example: Gaussians

The standard Gaussian $\mathcal{N}(0,1)$ does not have a closed form for its quantile function, but there are fast and accurate ways of calculating it numerically (they're what stand behind `qnorm`), so the quantile method can be used. In practice, there are other transformation methods which are even faster, but rely on special tricks.

Since $Q_Z(U)$ has the same distribution function as Z, we can use the quantile method, as long as we can calculate Q_Z. Since Q_Z always exists, in principle this solves the problem. In practice, we need to calculate Q_Z before we can use it, and this may not have a closed form, and numerical approximations may be intractable.[3] In such situations, we turn to more advanced methods (see further reading).

5.2.3 Sampling

A complement to drawing from given distributions is to **sample** from a given collection of objects. This is a common task, so R has a function to do it:

[3] In essence, we have to solve the nonlinear equation $F_Z(z) = p$ for z over and over for different p — and that assumes we can easily calculate F_Z.

```
sample(x, size, replace = FALSE, prob = NULL)
```

Here `x` is a vector which contains the objects we're going to sample from. `size` is the number of samples we want to draw from `x`. `replace` says whether the samples are drawn with or without replacement. (If `replace=TRUE`, then `size` can be arbitrarily larger than the length of `x`. If `replace=FALSE`, having a larger `size` doesn't make sense.) Finally, the optional argument `prob` allows for *weighted* sampling; ideally, `prob` is a vector of probabilities as long as `x`, giving the probability of drawing each element of `x`[4].

As a convenience for a common situation, running `sample` with one argument produces a random permutation of the input, i.e.,

```
sample(x)
```

is equivalent to

```
sample(x, size = length(x), replace = FALSE)
```

For example, the code for k-fold cross-validation, Code Example 3, had the lines

```
fold.labels <- sample(rep(1:nfolds, length.out = nrow(data)))
```

Here, `rep` repeats the numbers from 1 to `nfolds` until we have one number for each row of the data frame, say $1, 2, 3, 4, 5, 1, 2, 3, 4, 5, 1, 2$ if there were twelve rows. Then `sample` shuffles the order of those numbers randomly. This then would give an assignment of each row of `df` to one (and only one) of five folds.

5.2.3.1 Sampling Rows from Data Frames

When we have multivariate data (which is the usual situation), we typically arrange it into a data-frame, where each row records one unit of observation, with multiple interdependent columns. The natural notion of sampling is then to draw a random sample of the data points, which in that representation amounts to a random sample of the rows. We can implement this simply by sampling row *numbers*. For instance, this command,

```
df[sample(1:nrow(df), size = b), ]
```

will create a new data frame from `b`, by selecting `b` rows from `df` without replacement. It is an easy exercise to figure out how to sample from a data frame *with* replacement, and with unequal probabilities per row.

5.2.3.2 Multinomials and Multinoullis

If we want to draw one value from a multinomial distribution with probabilities $p = (p_1, p_2, \ldots p_k)$, then we can use sample:

[4] If the elements of `prob` do not add up to 1, but are positive, they will be normalized by their sum, e.g., setting `prob=c(9,9,1)` will assign probabilities $(\frac{9}{19}, \frac{9}{19}, \frac{1}{19})$ to the three elements of `x`.

```
sample(1:k, size = 1, prob = p)
```

If we want to simulate a "multinoulli" process[5], i.e., a sequence of independent and identically distributed multinomial random variables, then we can easily do so:

```
rmultinoulli <- function(n, prob) {
    k <- length(prob)
    return(sample(1:k, size = n, replace = TRUE, prob = prob))
}
```

Of course, the labels needn't be the integers $1:k$ (exercise 5.1).

5.2.3.3 Probabilities of Observation

Often, our models of how the data are generated will break up into two parts. One part is a model of how actual variables are related to each other out in the world. (E.g., we might model how education and racial categories are related to occupation, and occupation is related to income.) The other part is a model of how variables come to be recorded in our data, and the distortions they might undergo in the course of doing so. (E.g., we might model the probability that someone appears in a survey as a function of race and income.) Plausible sampling mechanisms often make the probability of appearing in the data a function of some of the variables. This can then have important consequences when we try to draw inferences about the whole population or process from the sample we happen to have seen (see, e.g., App. I).

```
income <- rnorm(n, mean = predict(income.model, x), sd = sigma)
capture.probabilities <- predict(observation.model, x)
observed.income <- sample(income, size = b, prob = capture.probabilities)
```

5.3 Repeating Simulations

Because simulations are often most useful when they are repeated many times, R has a command to repeat a whole block of code:

```
replicate(n, expr)
```

Here `expr` is some executable "expression" in R, basically something you could type in the terminal, and `n` is the number of times to repeat it.

For instance,

```
output <- replicate(1000, rnorm(length(x), beta0 + beta1 * x, sigma))
```

will replicate, 1000 times, sampling from the predictive distribution of a Gaussian linear regression model. Conceptually, this is equivalent to doing something like

[5] A handy term I learned from Gustavo Lacerda.

```
output <- matrix(0, nrow = 1000, ncol = length(x))
for (i in 1:1000) {
    output[i, ] <- rnorm(length(x), beta0 + beta1 * x, sigma)
}
```

but the `replicate` version has two great advantages. First, it is faster, because R processes it with specially-optimized code. (Loops are especially slow in R.) Second, and *far* more importantly, it is *clearer*: it makes it obvious what is being done, in one line, and leaves the computer to figure out the boring and mundane details of how best to implement it.

5.4 Why Simulate?

There are three major uses for simulation: to understand a model, to check it, and to fit it. We will deal with the first two here, and return to fitting in Chapter 24, after we've looked at dealing with dependence and hidden variables.

5.4.1 Understanding the Model; Monte Carlo

We understand a model by seeing what it predicts about the variables we care about, and the relationships between them. Sometimes those predictions are easy to extract from a mathematical representation of the model, but often they aren't. With a model we can simulate, however, we can just run the model and see what happens.

Our stochastic model gives a distribution for some random variable Z, which in general is a complicated, multivariate object with lots of interdependent components. We may also be interested in some complicated function g of Z, such as, say, the ratio of two components of Z, or even some nonparametric curve fit through the data points. How do we know what the model says about g?

Assuming we can make draws from the distribution of Z, we can find the distribution of any function of it we like, to as much precision as we want. Suppose that $\tilde{Z}_1, \tilde{Z}_2, \ldots \tilde{Z}_b$ are the outputs of b independent runs of the model — b different *replicates* of the model. (The tilde is a reminder that these are just simulations.) We can calculate g on each of them, getting $g(\tilde{Z}_1), g(\tilde{Z}_2), \ldots g(\tilde{Z}_b)$. If averaging makes sense for these values, then

$$\frac{1}{b}\sum_{i=1}^{b} g(\tilde{Z}_i) \xrightarrow[b\to\infty]{} \mathbb{E}\left[g(Z)\right] < \tag{5.7}$$

by the law of large numbers. So simulation and averaging lets us get expectation values. This basic observation is the seed of the **Monte Carlo method**.[6] If our

[6] The name was coined by the physicists at Los Alamos who used the method to do calculations relating to designing the hydrogen bomb; see Metropolis *et al.* (1953). (Folklore specifically credits it to Stanislaw Ulam, as a joking reference to the famous casino at the town of Monte Carlo in the principality of Monaco, on the French Riviera.) The technique was pioneered by the great physicist Enrico Fermi, who began using it in 1935 to do calculations relating to nuclear fission, but using

simulations are independent[7], we can even use the central limit theorem to say that $\frac{1}{b}\sum_{i=1}^{b} g(\tilde{Z}_i)$ has approximately the distribution $\mathcal{N}(\mathbb{E}[g(Z)], \mathbb{V}[g(Z)]/b)$. Of course, if you can get expectation values, you can also get variances. (This is handy if trying to apply the central limit theorem!) You can also get any higher moments — if, for whatever reason, you need the kurtosis, you just have to simulate enough.

You can also pick any set s and get the probability that $g(Z)$ falls into that set:

$$\frac{1}{b}\sum_{i=1}^{b} \mathbf{1}_s(g(\tilde{Z}_i)) \xrightarrow[b\to\infty]{} \Pr(g(Z) \in s) \tag{5.8}$$

The reason this works is of course that $\Pr(g(Z) \in s) = \mathbb{E}[\mathbf{1}_s(g(Z))]$, and we can use the law of large numbers again. So we can get the whole distribution of any complicated function of the model that we want, as soon as we can simulate the model. It is really only a little harder to get the complete sampling distribution than it is to get the expectation value, and the exact same ideas apply.

5.4.2 Checking the Model

An important but under-appreciated use for simulation is to *check* models after they have been fit. If the model is right, after all, it represents the mechanism which generates the data. This means that when we simulate, we run that mechanism, and the surrogate data which comes out of the machine should look like the real data. More exactly, the real data should look like a typical realization of the model. If it does not, then the model's account of the data-generating mechanism is systematically wrong in some way. By carefully choosing the simulations we perform, we can learn a lot about how the model breaks down and how it might need to be improved.[8]

5.4.2.1 "Exploratory" Analysis of Simulations

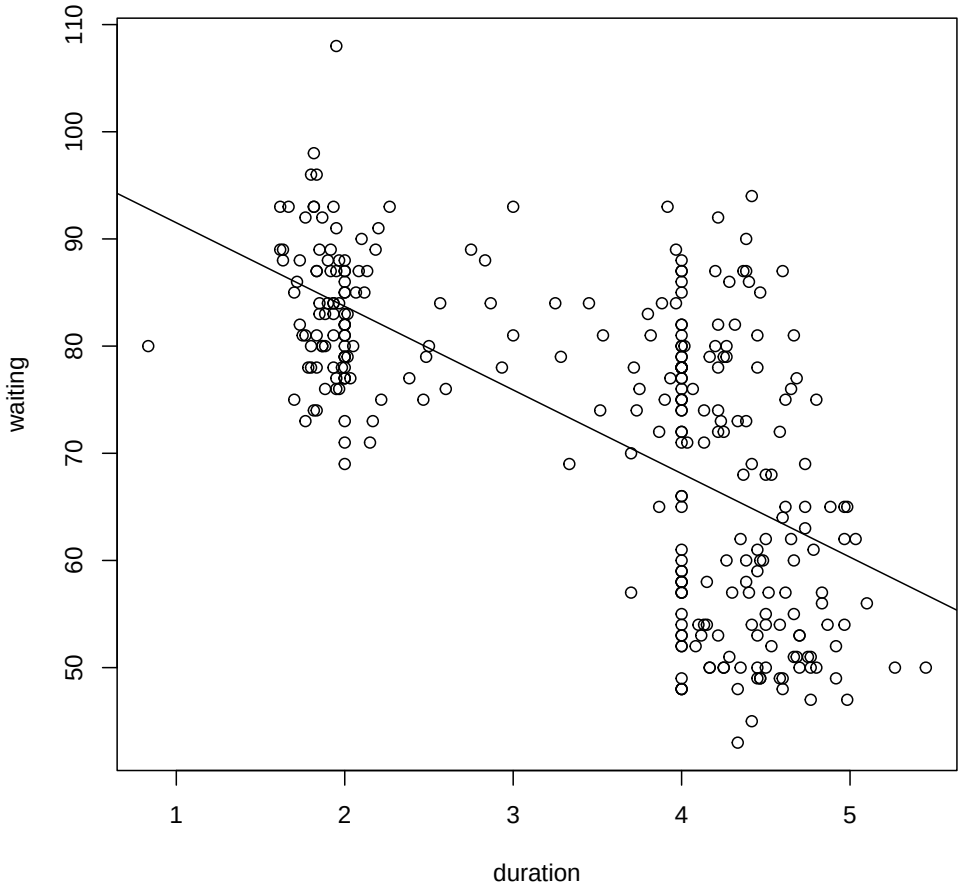
Often the comparison between simulations and data can be done qualitatively and visually. For example, a classic data set concerns the time between eruptions of the Old Faithful geyser in Yellowstone, and how they relate to the duration of the latest eruption. A common exercise is to fit a regression line to the data by ordinary least squares:

```
library(MASS)
data(geyser)
fit.ols <- lm(waiting ~ duration, data = geyser)
```

pencil, paper, and printed tables of random numbers, because programmable electronic computers did not exist yet (Schwartz, 2017, p. 124).

[7] Often our simulations are *dependent*, particularly in Markov chain Monte Carlo (MCMC), but there are still applicable central limit theorems. This is outside the scope of this chapter, but see the further reading.

[8] "Might", because sometimes (e.g., §1.4.2) we're better off with a model that makes systematic mistakes, if they're small and getting it right would be a hassle.




```
plot(geyser$duration, geyser$waiting, xlab = "duration", ylab = "waiting")
abline(fit.ols)
```

Figure 5.1 Data for the `geyser` data set, plus the OLS regression line.

Figure 5.1 shows the data, together with the OLS line. It doesn't look that great, but if someone insisted it was a triumph of quantitative vulcanology, how could you show they were wrong?

We'll consider general tests of regression specifications in Chapter 9. For now, let's focus on the way OLS is usually presented as part of a stochastic model for the response conditional on the input, with Gaussian and homoskedastic noise. In this case, the stochastic model is $\texttt{waiting} = \beta_0 + \beta_1 \texttt{duration} + \epsilon$, with $\epsilon \sim \mathcal{N}(0, \sigma^2)$. If we simulate from this probability model, we'll get something we can

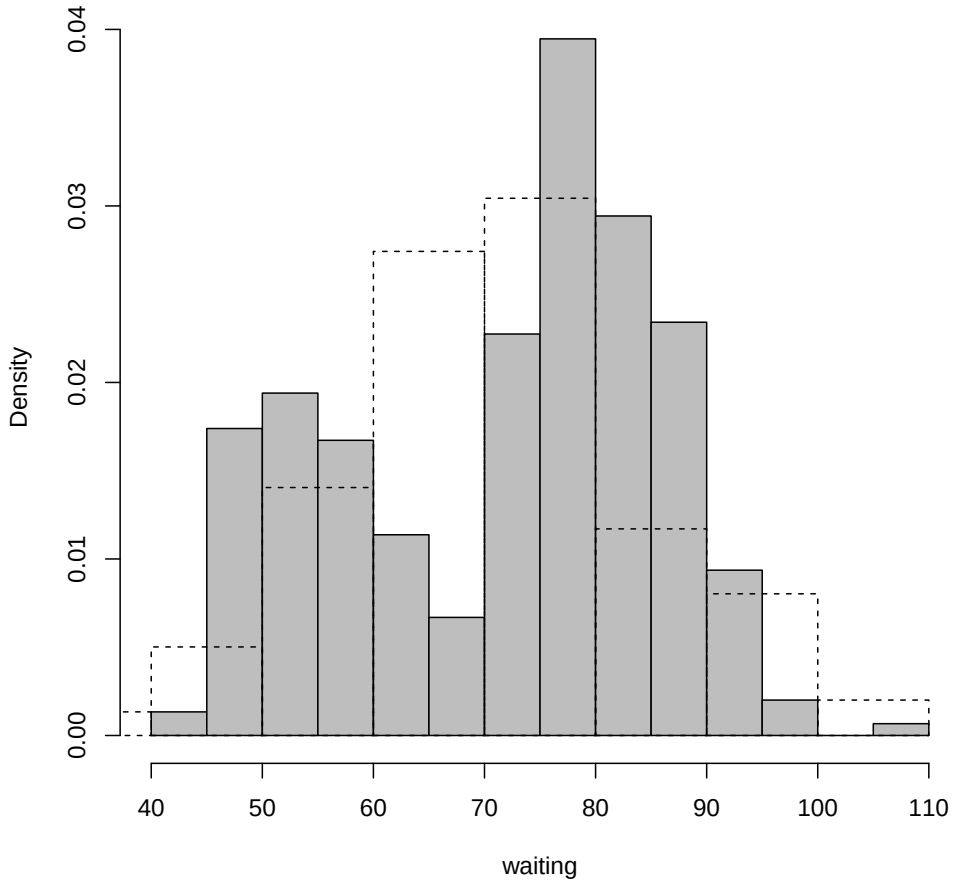
```
rgeyser <- function() {
    n <- nrow(geyser)
    sigma <- summary(fit.ols)$sigma
    new.waiting <- rnorm(n, mean = fitted(fit.ols), sd = sigma)
    new.geyser <- data.frame(duration = geyser$duration, waiting = new.waiting)
    return(new.geyser)
}
```

CODE EXAMPLE 6: *Function for generating surrogate data sets from the linear model fit to* `geyser`.

compare to the actual data, to help us assess whether the scatter around that regression line is really bothersome. Since OLS doesn't require us to assume a distribution for the input variable (here, `duration`), the simulation function in Code Example 6 leaves those values alone, but regenerates values of the response (`waiting`) according to the model assumptions.

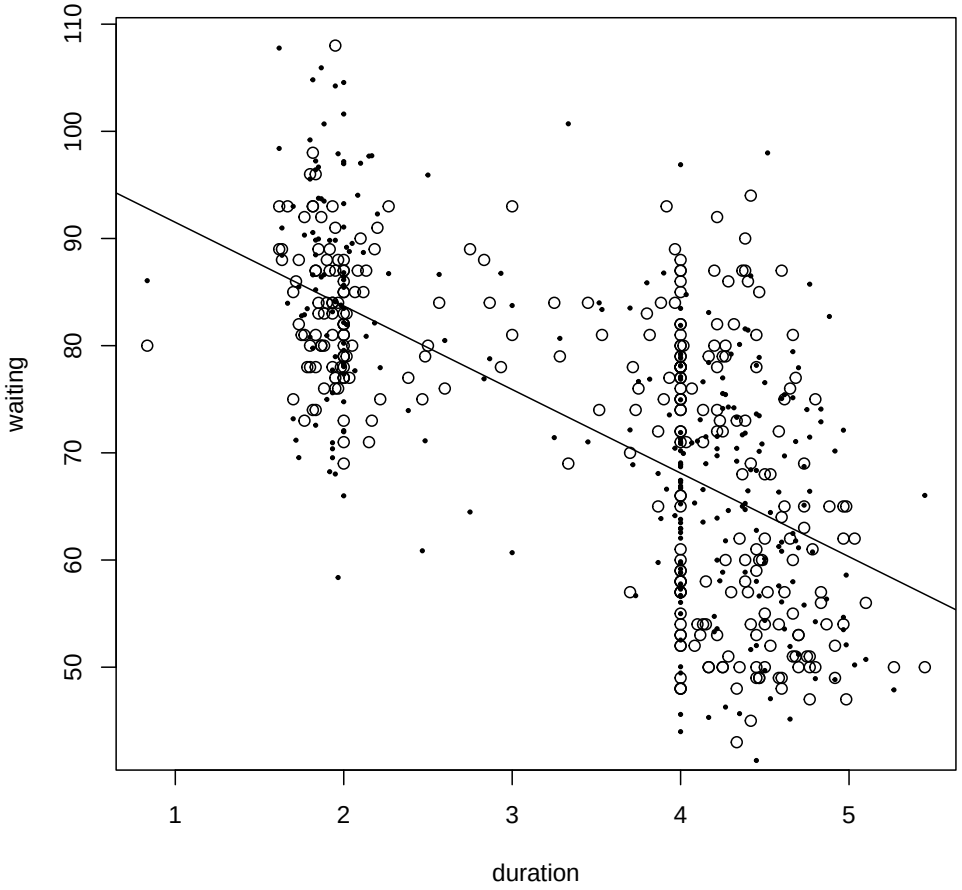
A useful principle for model checking is that if we do some exploratory data analyses of the real data, doing the same analyses to realizations of the model should give roughly the same results (Gelman, 2003; Hunter *et al.*, 2008; Gelman and Shalizi, 2013). This is a test the model fails. Figure 5.2 shows the actual histogram of `waiting`, plus the histogram produced by simulating — reality is clearly bimodal, but the model is unimodal. Similarly, Figure 5.3 shows the real data, the OLS line, and a simulation from the OLS model. It's visually clear that the deviations of the real data from the regression line are both bigger and more patterned than those we get from simulating the model, so something is wrong with the latter.

By itself, just seeing that data doesn't look like a realization of the model isn't super informative, since we'd really like to know *how* the model's broken, and so how to fix it. Further simulations, comparing more detailed analyses of the data to analyses of the simulation output, are often very helpful here. Looking at Figure 5.3, we might suspect that one problem is heteroskedasticity — the variance isn't constant. This suspicion is entirely correct, and will be explored in §10.3.2.




```
hist(geyser$waiting, freq = FALSE, xlab = "waiting", main = "", sub = "", col = "grey")
lines(hist(rgeyser()$waiting, plot = FALSE), freq = FALSE, lty = "dashed")
```

Figure 5.2 Actual density of the waiting time between eruptions (grey bars, solid lines) and that produced by simulating the OLS model (dashed lines).




```
plot(geyser$duration, geyser$waiting, xlab = "duration", ylab = "waiting")
abline(fit.ols)
points(rgeyser(), pch = 20, cex = 0.5)
```

Figure 5.3 As in Figure 5.1, plus one realization of simulating the OLS model (small black dots).

5.4.3 Sensitivity Analysis

Often, the statistical inference we do on the data is predicated on certain assumptions about how the data is generated. We've talked a lot about the Gaussian-noise assumptions that usually accompany linear regression, but there are many others. For instance, if we have missing values for some variables and just ignore incomplete rows, we are implicitly assuming that data are "missing at random", rather than in some systematic way that would carry information about *what* the missing values were (see App. I). Often, these assumptions make our analysis much neater than it otherwise would be, so it would be *convenient* if they were true.

As a wise man said long ago, "The method of 'postulating' what we want has many advantages; they are the same as the advantages of theft over honest toil" (Russell, 1920, ch. VII, p. 71). In statistics, honest toil often takes the form of **sensitivity analysis**, of seeing how much our conclusions would change if the assumptions were violated, i.e., of checking how sensitive our inferences are to the assumptions. In principle, this means setting up models where the assumptions are more or less violated, or violated in different ways, analyzing them as though the assumptions held, and seeing how badly wrong we go. Of course, if that was easy to do in closed form, we often wouldn't have needed to make those assumptions in the first place.

On the other hand, it's usually pretty easy to *simulate* a model where the assumption is violated, run our original, assumption-laden analysis on the simulation output, and see what happens. Because it's a simulation, we know the complete truth about the data-generating process, and can assess how far off our inferences are. In favorable circumstances, our inferences don't mess up too much even when the assumptions we used to motivate the analysis are badly wrong. Sometimes, however, we discover that even tiny violations of our initial assumptions lead to large errors in our inferences. Then we either need to make some compelling case for those assumptions, or be *very* cautious in our inferences.

5.5 Further Reading

Simulation will be used in nearly every subsequent chapter. It is the key to the "bootstrap" technique for quantifying uncertainty (Ch. 6), and the foundation for a whole set of methods for dealing with complex models of dependent data (Ch. 24).

Many texts on scientific programming discuss simulation, including Press *et al.* (1992) and, using R, Jones *et al.* (2009). There are also many more specialized texts on simulation in various applied areas. It must be said that many references on simulation present it as almost completely disconnected from statistics and data analysis, giving the impression that probability models just fall from the sky. Guttorp (1995) is an excellent exception.

Random-variable generation is a standard topic in computational statistics, so there are lots of perfectly decent references, e.g., Press *et al.* (1992) or Monahan

(2001); at a higher level of technicality, Devroye (1986) is authoritative. Many of these references also cover methods of generating uniformly distributed (pseudo-)random numbers as a fundamental input.

On Monte Carlo: Robert and Casella (2004) is a standard authority on applications and techniques common in statistics. It has particularly good coverage of the important technique of Markov chain Monte Carlo, which is used when it's easier to get many *dependent* samples from the desired distribution than independent ones. Newman and Barkema (1999) is excellent if you know some physics, especially thermodynamics.

When all (!) you need to do is draw numbers from a probability distribution which isn't one of the ones built in to R, it's worth checking CRAN's "task view" on probability distributions, `https://cran.r-project.org/web/views/Distributions.html`.

For sensitivity analyses, Miller (1998) describes how to use modern optimization methods to actively search for settings in simulation models which break desired behaviors or conclusions. I have not seen this idea applied to sensitivity analyses for statistical models, but it really ought to be.

Exercises

5.1 Modify `rmultinoulli` from §5.2.3.2 so that the values in the output are not the integers from 1 to k, but come from a vector of arbitrary labels.

6

The Bootstrap

We are now several chapters into a statistics class and have said basically nothing about uncertainty. This should seem odd, and may even be disturbing if you are very attached to your p-values and saying variables have "significant effects". It is time to remedy this, and talk about how we can quantify uncertainty for complex models. The key technique here is what's called **bootstrapping**, or **the bootstrap**.

6.1 Stochastic Models, Uncertainty, Sampling Distributions

Statistics is the branch of mathematical engineering which studies ways of drawing inferences from limited and imperfect data. We want to know how a neuron in a rat's brain responds when one of its whiskers gets tweaked, or how many rats live in Pittsburgh, or how high the water will get under the 16th Street bridge during May, or the typical course of daily temperatures in the city over the year, or the relationship between the number of birds of prey in Schenley Park in the spring and the number of rats the previous fall. We have some data on all of these things. But we know that our data is incomplete, and experience tells us that repeating our experiments or observations, even taking great care to replicate the conditions, gives more or less different answers every time. It is foolish to treat any inference from the data in hand as certain.

If all data sources were totally capricious, there'd be nothing to do beyond piously qualifying every conclusion with "but we could be wrong about this". A mathematical discipline of statistics is possible because while repeating an experiment gives different results, some kinds of results are more common than others; their relative frequencies are reasonably stable. We thus model the data-generating mechanism through probability distributions and stochastic processes. When and why we can use stochastic models are very deep questions, but ones for another time. If we *can* use them in our problem, quantities like the ones I mentioned above are represented as functions of the stochastic model, i.e., of the underlying probability distribution. Since a function of a function is a "functional", and these quantities are functions of the true probability distribution function, we'll call these **functionals** or **statistical functionals**[1]. Functionals could be single numbers (like the total rat population), or vectors, or even whole

[1] Most writers in theoretical statistics just call them "parameters" in a generalized sense, but I will

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

curves (like the expected time-course of temperature over the year, or the regression of hawks now on rats earlier). Statistical inference becomes estimating those functionals, or testing hypotheses about them.

These estimates and other inferences are functions of the data values, which means that they inherit variability from the underlying stochastic process. If we "re-ran the tape" (as the late, great Stephen Jay Gould used to say), we would get different data, with a certain characteristic distribution, and applying a fixed procedure would yield different inferences, again with a certain distribution. Statisticians want to use this distribution to quantify the uncertainty of the inferences. For instance, the standard error is an answer to the question "By how much would our estimate of this functional vary, typically, from one replication of the experiment to another?" (It presumes a particular meaning for "typically vary", as the root-mean-square deviation around the mean.) A confidence region on a parameter, likewise, is the answer to "What are all the values of the parameter which *could* have produced this data with at least some specified probability?", i.e., all the parameter values under which our data are not low-probability outliers. The confidence region is a promise that *either* the true parameter point lies in that region, *or* something very unlikely under any circumstances happened — or that our stochastic model is wrong.

To get things like standard errors or confidence intervals, we need to know the distribution of our estimates around the true values of our functionals. These **sampling distributions** follow, remember, from the distribution of the data, since our estimates are functions of the data. Mathematically the problem is well-defined, but actually *computing* anything is another story. Estimates are typically complicated functions of the data, and mathematically-convenient distributions may all be poor approximations to the data source. Saying anything in closed form about the distribution of estimates can be simply hopeless. The two classical responses of statisticians were to focus on tractable special cases, and to appeal to asymptotics.

Your introductory statistics courses mostly drilled you in the special cases. From one side, limit the kind of estimator we use to those with a simple mathematical form — say, means and other linear functions of the data. From the other, assume that the probability distributions featured in the stochastic model take one of a few forms for which exact calculation *is* possible, analytically or via tabulated special functions. Most such distributions have origin myths: the Gaussian arises from averaging many independent variables of equal size (say, the many genes which contribute to height in humans); the Poisson distribution comes from counting how many of a large number of independent and individually-improbable events have occurred (say, radioactive nuclei decaying in a given second), etc. Squeezed from both ends, the sampling distribution of estimators and other functions of the data becomes exactly calculable in terms of the aforementioned special functions.

try to restrict that word to actual parameters specifying statistical models, to minimize confusion. I may slip up.

That these origin myths invoke various limits is no accident. The great results of probability theory — the laws of large numbers, the ergodic theorem, the central limit theorem, etc. — describe limits in which *all* stochastic processes in broad classes of models display the same asymptotic behavior. The central limit theorem, for instance, says that if we average more and more independent random quantities with a common distribution, and that common distribution isn't too pathological, then the average becomes closer and closer to a Gaussian[2]. Typically, as in the CLT, the limits involve taking more and more data from the source, so statisticians use the theorems to find the asymptotic, large-sample distributions of their estimates. We have been especially devoted to re-writing our estimates as averages of independent quantities, so that we can use the CLT to get Gaussian asymptotics.

Up through about the 1960s, statistics was split between developing general ideas about how to draw and evaluate inferences with stochastic models, and working out the properties of inferential procedures in tractable special cases (especially the linear-and-Gaussian case), or under asymptotic approximations. This yoked a very broad and abstract theory of inference to very narrow and concrete practical formulas, an uneasy combination often preserved in basic statistics classes.

The arrival of (comparatively) cheap and fast computers made it feasible for scientists and statisticians to record lots of data and to fit models to it, so they did. Sometimes the models were conventional ones, including the special-case assumptions, which often enough turned out to be detectably, and consequentially, wrong. At other times, scientists wanted more complicated or flexible models, some of which had been proposed long before, but now moved from being theoretical curiosities to stuff that could run overnight[3]. In principle, asymptotics might handle either kind of problem, but convergence to the limit could be unacceptably slow, especially for more complex models.

By the 1970s, then, statistics faced the problem of quantifying the uncertainty of inferences without using either implausibly-helpful assumptions or asymptotics; all of the solutions turned out to demand *even more* computation. Here we will examine what may be the most successful solution, Bradley Efron's proposal to combine estimation with simulation, which he gave the less-than-clear but persistent name of "the bootstrap" (Efron, 1979).

6.2 The Bootstrap Principle

Remember (from baby stats.) that the key to dealing with uncertainty in parameters and functionals is the sampling distribution of estimators. Knowing what distribution we'd get for our estimates on repeating the experiment would give us things like standard errors. Efron's insight was that we can *simulate* repli-

[2] The reason is that the non-Gaussian parts of the distribution wash away under averaging, but the average of two Gaussians is another Gaussian.

[3] Kernel regression (§1.5.2), kernel density estimation (Ch. 14), and nearest-neighbors prediction (§1.5.1) were all proposed in the 1950s or 1960s, but didn't begin to be widely used until about 1980.

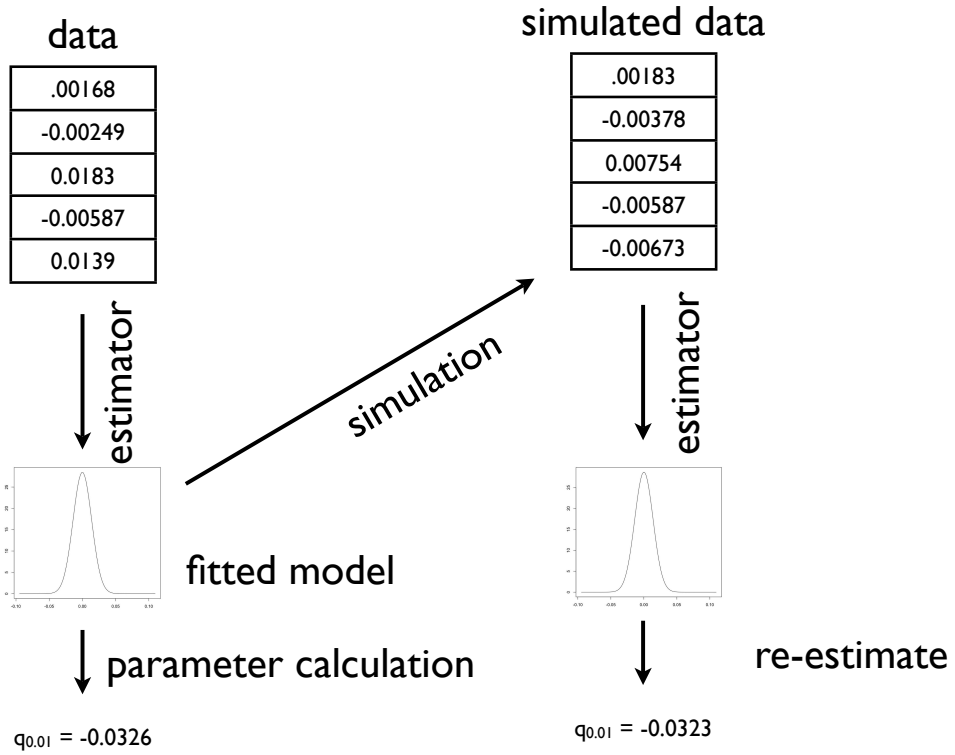


Figure 6.1 Schematic for model-based bootstrapping: simulated values are generated from the fitted model, then treated like the original data, yielding a new estimate of the functional of interest, here called $q_{0.01}$.

cation. After all, we have already fitted a model to the data, which is a guess at the mechanism which generated the data. Running that mechanism generates simulated data which, by hypothesis, has the same distribution as the real data. Feeding the simulated data through our estimator gives us one draw from the sampling distribution; repeating this many times yields the sampling distribution. Since we are using the model to give us its own uncertainty, Efron called this "bootstrapping"; unlike the Baron Munchhausen's plan for getting himself out of a swamp by pulling on his own bootstraps, it works.

Figure 6.1 sketches the over-all process: fit a model to data, use the model to calculate the functional, then get the sampling distribution by generating new, synthetic data from the model and repeating the estimation on the simulation output.

To fix notation, we'll say that the original data is x. (In general this is a whole data frame, not a single number.) Our parameter estimate from the data is $\hat{\theta}$. Surrogate data sets simulated from the fitted model will be $\tilde{X}_1, \tilde{X}_2, \ldots \tilde{X}_B$. The corresponding re-estimates of the parameters on the surrogate data are $\tilde{\theta}_1, \tilde{\theta}_2, \ldots \tilde{\theta}_B$.

The functional of interest is estimated by the statistic[4] T, with sample value $\hat{t} = T(x)$, and values of the surrogates of $\tilde{t}_1 = T(\tilde{X}_1)$, $\tilde{t}_2 = T(\tilde{X}_2)$, $\ldots \tilde{t}_B = T(\tilde{X}_B)$. (The statistic T may be a direct function of the estimated parameters, and only indirectly a function of x.) Everything which follows applies without modification when the functional of interest *is* the parameter, or some component of the parameter.

In this section, we will assume that the model is correct for *some* value of θ, which we will call θ_0. This means that we are employing a parametric **model-based bootstrap**. The true (population or ensemble) values of the functional is likewise t_0.

6.2.1 Variances and Standard Errors

The simplest thing to do is to get the variance or standard error:

$$\widehat{\mathrm{Var}}\left[\hat{t}\right] = \mathbb{V}\left[\tilde{t}\right] \tag{6.1}$$

$$\widehat{\mathrm{se}}(\hat{t}) = \mathrm{sd}(\tilde{t}) \tag{6.2}$$

That is, we approximate the variance of our estimate of t_0 under the true but unknown distribution θ_0 by the variance of re-estimates $\tilde{t}$ on surrogate data from the fitted model $\widehat{\theta}$. Similarly we approximate the true standard error by the standard deviation of the re-estimates. The logic here is that the simulated $\tilde{X}$ has about the same distribution as the real X that our data, x, was drawn from, so applying the same estimation procedure to the surrogate data gives us the sampling distribution. This assumes, of course, that our model is right, and that $\hat{\theta}$ is not too far from θ_0.

A code sketch is provided in Code Example 7. Note that this may not work *exactly* as given in some circumstances, depending on the syntax details of, say, just what kind of data structure is needed to store $\hat{t}$.

[4] T is a common symbol in the literature on the bootstrap for a *generic* function of the data. It may or may not have anything to do with Student's t test for difference in means.

```
rboot <- function(statistic, simulator, B) {
    tboots <- replicate(B, statistic(simulator()))
    if (is.null(dim(tboots))) {
        tboots <- array(tboots, dim = c(1, B))
    }
    return(tboots)
}

bootstrap <- function(tboots, summarizer, ...) {
    summaries <- apply(tboots, 1, summarizer, ...)
    return(t(summaries))
}

bootstrap.se <- function(statistic, simulator, B) {
    bootstrap(rboot(statistic, simulator, B), summarizer = sd)
}
```

CODE EXAMPLE 7: *Code for calculating bootstrap standard errors. The function* `rboot` *generates* `B` *bootstrap samples (using the* `simulator` *function) and calculates the statistic on them (using* `statistic`*).* `simulator` *needs to be a function which returns a surrogate data set in a form suitable for* `statistic`*. (How would you modify the code to pass arguments to* `simulator` *and/or* `statistic`*?) Because every use of bootstrapping is going to need to do this, it makes sense to break it out as a separate function, rather than writing the same code many times (with many chances of getting it wrong). The* `bootstrap` *function takes the output of* `rboot` *and applies a summarizing function.* `bootstrap.se` *just calls* `rboot` *and makes the summarizing function* `sd`*, which takes a standard deviation.* IMPORTANT NOTE: *This is just a code* sketch*, because depending on the data structure which the statistic returns, it may not (e.g.) be feasible to just run* `sd` *on it, and so it might need some modification. See detailed examples below.*

```
bootstrap.bias <- function(simulator, statistic, B, t.hat) {
    expect <- bootstrap(rboot(statistic, simulator, B), summarizer = mean)
    return(expect - t.hat)
}
```

CODE EXAMPLE 8: *Sketch of code for bootstrap bias correction. Arguments are as in Code Example 7, except that* `t.hat` *is the estimate on the original data.* IMPORTANT NOTE: *As with Code Example 7, this is just a code* sketch*, because it won't work with all data types that might be returned by* `statistic`*, and so might require modification.*

6.2.2 Bias Correction

We can use bootstrapping to correct for a biased estimator. Since the sampling distribution of $\tilde{t}$ is close to that of $\widehat{t}$, and $\widehat{t}$ itself is close to t_0,

$$\mathbb{E}\left[\widehat{t}\right] - t_0 \approx \mathbb{E}\left[\tilde{t}\right] - \widehat{t} \tag{6.3}$$

The left hand side is the bias that we want to know, and the right-hand side the was what we can calculate with the bootstrap.

In fact, Eq. 6.3 remains valid so long as the sampling distribution of $\widehat{t} - t_0$ is close to that of $\tilde{t} - \widehat{t}$. This is a weaker requirement than asking for $\widehat{t}$ and $\tilde{t}$ themselves to have similar distributions, or asking for $\widehat{t}$ to be close to t_0. In statistical theory, a random variable whose distribution does not depend on the parameters is called a **pivot**. (The metaphor is that it stays in one place while the parameters turn around it.) A sufficient (but not necessary) condition for Eq. 6.3 to hold is that $\widehat{t} - t_0$ be a pivot, or approximately pivotal.

6.2.3 Confidence Intervals

A confidence interval is a random interval which contains the truth with high probability (the confidence level). If the confidence interval for g is C, and the confidence level is $1-\alpha$, then we want

$$\Pr(t_0 \in C) = 1-\alpha \tag{6.4}$$

no matter what the true value of t_0. When we calculate a confidence interval, our inability to deal with distributions exactly means that the true confidence level, or **coverage** of the interval, is not quite the desired confidence level $1-\alpha$; the closer it is, the better the approximation, and the more accurate the confidence interval.[5]

When we simulate, we get samples of $\tilde{t}$, but what we really care about is the distribution of $\hat{t}$. When we have enough data to start with, those two distributions will be approximately the same. But at any given amount of data, the distribution of $\tilde{t}-\hat{t}$ will usually be closer to that of $\hat{t}-t_0$ than the distribution of $\tilde{t}$ is to that of $\hat{t}$. That is, the distribution of fluctuations around the true value usually converges quickly. (Think of the central limit theorem.) We can use this to turn information about the distribution of $\tilde{t}$ into accurate confidence intervals for t_0, essentially by re-centering $\tilde{t}$ around $\hat{t}$.

Specifically, let $q_{\alpha/2}$ and $q_{1-\alpha/2}$ be the $\alpha/2$ and $1-\alpha/2$ quantiles of $\tilde{t}$. Then

$$1-\alpha = \Pr\left(q_{\alpha/2} \leq \tilde{T} \leq q_{1-\alpha/2}\right) \tag{6.5}$$

$$= \Pr\left(q_{\alpha/2} - \hat{T} \leq \tilde{T} - \hat{T} \leq q_{1-\alpha/2} - \hat{T}\right) \tag{6.6}$$

$$\approx \Pr\left(q_{\alpha/2} - \hat{T} \leq \hat{T} - t_0 \leq q_{1-\alpha/2} - \hat{T}\right) \tag{6.7}$$

$$= \Pr\left(q_{\alpha/2} - 2\hat{T} \leq -t_0 \leq q_{1-\alpha/2} - 2\hat{T}\right) \tag{6.8}$$

$$= \Pr\left(2\hat{T} - q_{1-\alpha/2} \leq t_0 \leq 2\hat{T} - q_{\alpha/2}\right) \tag{6.9}$$

The interval $C = [2\hat{T} - q_{\alpha/2}, 2\hat{T} - q_{1-\alpha/2}]$ is random, because $\hat{T}$ is a random quantity, so it makes sense to talk about the probability that it contains the true value t_0. Also, notice that the upper and lower quantiles of $\tilde{T}$ have, as it were, swapped roles in determining the upper and lower confidence limits. Finally, notice that we do not actually know those quantiles exactly, but they're what we approximate by bootstrapping.

This is the **basic bootstrap confidence interval**, or the **pivotal** CI. It is simple and reasonably accurate, and makes a very good default choice for finding confidence intervals.

[5] You might wonder why we'd be unhappy if the coverage level was *greater* than $1-\alpha$. This is certainly better than if it's *less* than the nominal confidence level, but it usually means we could have used a smaller set, and so been more precise about t_0, without any more real risk. Confidence intervals whose coverage is greater than the nominal level are called **conservative**; those with less than nominal coverage are **anti-conservative** (and not, say, "liberal").

```
equitails <- function(x, alpha) {
    lower <- quantile(x, alpha/2)
    upper <- quantile(x, 1 - alpha/2)
    return(c(lower, upper))
}

bootstrap.ci <- function(statistic = NULL, simulator = NULL, tboots = NULL, B = if (!is.null(tboots)
    ncol(tboots)
}, t.hat, level) {
    if (is.null(tboots)) {
        stopifnot(!is.null(statistic))
        stopifnot(!is.null(simulator))
        stopifnot(!is.null(B))
        tboots <- rboot(statistic, simulator, B)
    }
    alpha <- 1 - level
    intervals <- bootstrap(tboots, summarizer = equitails, alpha = alpha)
    upper <- t.hat + (t.hat - intervals[, 1])
    lower <- t.hat + (t.hat - intervals[, 2])
    CIs <- cbind(lower = lower, upper = upper)
    return(CIs)
}
```

Code Example 9: *Sketch of code for calculating the basic bootstrap confidence interval. See Code Example 7 for* `rboot` *and* `bootstrap`*, and cautions about blindly applying this to arbitrary data-types.*

6.2.3.1 Other Bootstrap Confidence Intervals

The basic bootstrap CI relies on the distribution of $\tilde{t} - \hat{t}$ being approximately the same as that of $\hat{t} - t_0$. Even when this is false, however, it can be that the distribution of

$$\tau = \frac{\hat{t} - t_0}{\widehat{\text{se}}(\hat{t})} \tag{6.10}$$

is close to that of

$$\tilde{\tau} = \frac{\tilde{t} - \hat{t}}{\text{se}(\tilde{t})} \tag{6.11}$$

This is like what we calculate in a t-test, and since the t-test was invented by "Student", these are called **studentized** quantities. If τ and $\tilde{\tau}$ have the same distribution, then we can reason as above and get a confidence interval

$$\left(\hat{t} - \widehat{\text{se}}(\hat{t}) Q_{\tilde{\tau}}(1-\alpha/2), \hat{t} - \widehat{\text{se}}(\hat{t}) Q_{\tilde{\tau}}(\alpha/2)\right) \tag{6.12}$$

This is the same as the basic interval when $\widehat{\text{se}}(\hat{t}) = \text{se}(\tilde{t})$, but different otherwise. To find $\text{se}(\tilde{t})$, we need to actually do a *second* level of bootstrapping, as follows.

1. Fit the model with $\hat{\theta}$, find $\hat{t}$.
2. For $i \in 1 : B_1$
    1. Generate $\tilde{X}_i$ from $\hat{\theta}$
    2. Estimate $\tilde{\theta}_i$, $\tilde{t}_i$
    3. For $j \in 1 : B_2$
        1. Generate $X_{ij}^\dagger$ from $\tilde{\theta}_i$
        2. Calculate $t_{ij}^\dagger$
    4. Set $\tilde{\sigma}_i$ = standard deviation of the $t_{ij}^\dagger$
    5. Set $\tilde{\tau}_{ij} = \frac{t_{ij}^\dagger - \tilde{t}_i}{\tilde{\sigma}_i}$ for all j
3. Set $\widehat{\text{se}}(\hat{t})$ = standard deviation of the $\tilde{t}_i$
4. Find the $\alpha/2$ and $1-\alpha/2$ quantiles of the distribution of the $\tilde{\tau}$
5. Plug into Eq. 6.12.

The advantage of the studentized intervals is that they are more accurate than the basic ones; the disadvantage is that they are more work! At the other extreme, the **percentile method** simply sets the confidence interval to

$$(Q_{\tilde{t}}(\alpha/2), Q_{\tilde{t}}(1-\alpha/2)) \tag{6.13}$$

This is definitely easier to calculate, but not as accurate as the basic, pivotal CI.

All of these methods have many variations, described in the monographs referred to at the end of this chapter (§6.9).

```
boot.pvalue <- function(test, simulator, B, testhat) {
    testboot <- rboot(B = B, statistic = test, simulator = simulator)
    p <- (sum(testboot >= testhat) + 1)/(B + 1)
    return(p)
}
```

CODE EXAMPLE 10: *Bootstrap p-value calculation.* `testhat` *should be the value of the test statistic on the actual data.* `test` *is a function which takes in a data set and calculates the test statistic, presuming that large values indicate departure from the null hypothesis. Note the* $+1$ *in the numerator and denominator of the p-value — it would be more straightforward to leave them off, but this is a little more stable when* B *is comparatively small. (Also, it keeps us from ever reporting a p-value of exactly 0.)*

6.2.4 Hypothesis Testing

For hypothesis tests, we may want to calculate two sets of sampling distributions: the distribution of the test statistic under the null tells us about the size of the test and significance levels, and the distribution under the alternative tells us about power and realized power. We can find either with bootstrapping, by simulating from either the null or the alternative. In such cases, the statistic of interest, which I've been calling T, is the test statistic. Code Example 10 illustrates how to find a p-value by simulating under the null hypothesis. The same procedure would work to calculate power, only we'd need to simulate from the alternative hypothesis, and `testhat` would be set to the critical value of T separating acceptance from rejection, not the observed value.

```
doubleboot.pvalue <- function(test, simulator, B1, B2, estimator, thetahat, testhat,
    ...) {
    for (i in 1:B1) {
        xboot <- simulator(theta = thetahat, ...)
        thetaboot <- estimator(xboot)
        testboot[i] <- test(xboot)
        pboot[i] <- boot.pvalue(test, simulator, B2, testhat = testboot[i], theta = thetaboot)
    }
    p <- (sum(testboot >= testhat) + 1)/(B1 + 1)
    p.adj <- (sum(pboot <= p) + 1)/(B1 + 1)
    return(p.adj)
}
```

CODE EXAMPLE 11: *Code sketch for "double bootstrap" significance testing. The inner or second bootstrap is used to calculate the distribution of nominal bootstrap p-values. For this to work, we need to draw our second-level bootstrap samples from $\tilde{\theta}$, the bootstrap re-estimate, not from $\hat{\theta}$, the data estimate. The code presumes the* `simulator` *function takes a* `theta` *argument allowing this. Exercise: replace the* `for` *loop with* `replicate`*.*

6.2.4.1 Double bootstrap hypothesis testing

When the hypothesis we are testing involves estimated parameters, we may need to correct for this. Suppose, for instance, that we are doing a goodness-of-fit test. If we estimate our parameters on the data set, we adjust our distribution so that it matches the data. It is thus not surprising if it seems to fit the data well! (Essentially, it's the problem of evaluating performance by looking at in-sample fit, which gave us so much trouble in Chapter 3.)

Some test statistics have distributions which are not affected by estimating parameters, at least not asymptotically. In other cases, one can analytically come up with correction terms. When these routes are blocked, one uses a **double bootstrap**, where a second level of bootstrapping checks how much estimation improves the apparent fit of the model. This is perhaps most easily explained in pseudo-code (Code Example 11).

6.2.5 Model-Based Bootstrapping Example: Pareto's Law of Wealth Inequality

The **Pareto** or **power-law** distribution[6], is a popular model for data with "heavy tails", i.e. where the probability density $f(x)$ goes to zero only very slowly as $x \to \infty$. The probability density is

$$f(x) = \frac{\theta - 1}{x_0}\left(\frac{x}{x_0}\right)^{-\theta} \tag{6.14}$$

where x_0 is the minimum scale of the distribution, and θ is the **scaling exponent** (Exercise 6.1). The Pareto is highly right-skewed, with the mean being much larger than the median.

If we know x_0, one can show that the maximum likelihood estimator of the exponent θ is

$$\hat{\theta} = 1 + \frac{n}{\sum_{i=1}^{n} \log \frac{x_i}{x_0}} \tag{6.15}$$

and that this is consistent (Exercise 6.3), and efficient. Picking x_0 is a harder problem (see Clauset *et al.* 2009) — for the present purposes, pretend that the Oracle tells us. The file `pareto.R`, on the book website, contains a number of functions related to the Pareto distribution, including a function `pareto.fit` for estimating it. (There's an example of its use below.)

Pareto came up with this density when he attempted to model the distribution of personal wealth. Approximately, but quite robustly across countries and time-periods, the upper tail of the distribution of income and wealth follows a power law, with the exponent varying as money is more or less concentrated among the very richest individuals and households[7]. Figure 6.2 shows the distribution of net worth for the 400 richest Americans in 2003[8].

```
source("http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/code/pareto.R")
wealth <- scan("http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/data/wealth.dat")
x0 <- 9e+08
n.tail <- sum(wealth >= x0)
wealth.pareto <- pareto.fit(wealth, threshold = x0)
```

Taking $x_0 = 9 \times 10^8$ (again, see Clauset *et al.* 2009), the number of individuals in the tail is 302, and the estimated exponent is $\hat{\theta} = 2.34$.

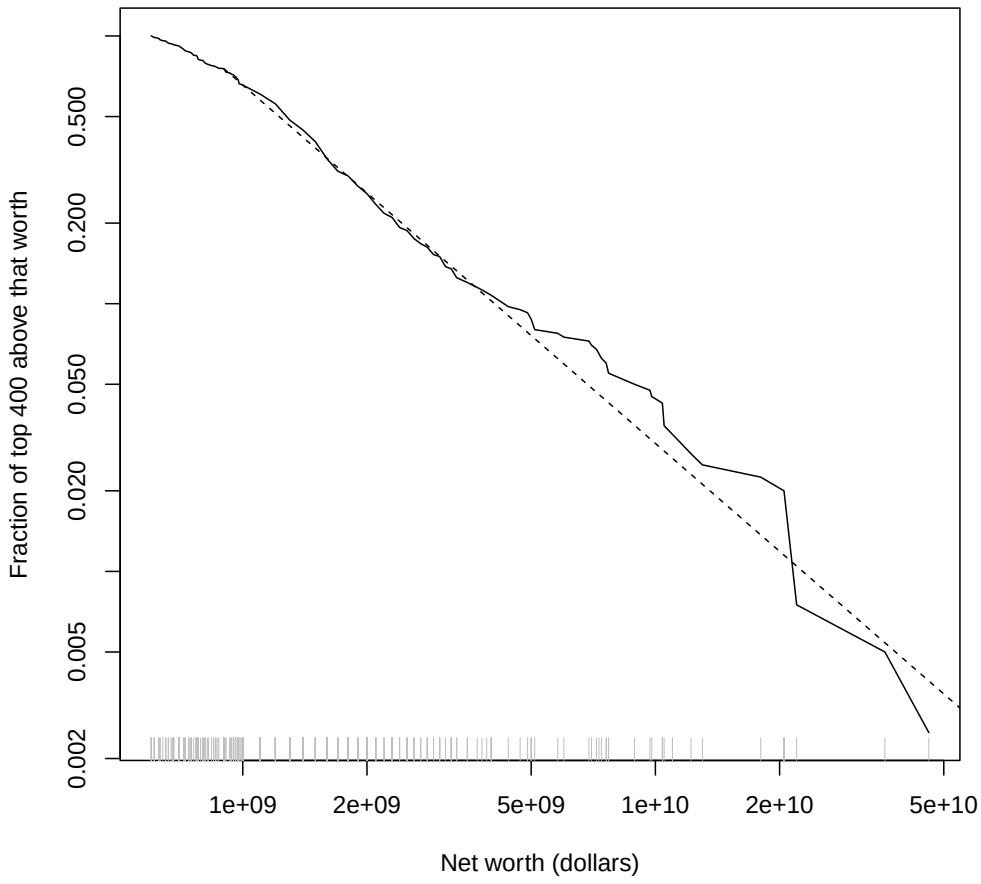
How much uncertainty is there in this estimate of the exponent? Naturally, we'll bootstrap. We need a function to generate Pareto-distributed random variables; this, along with some related functions, is part of the file `pareto.R` on the course website. With that tool, model-based bootstrapping proceeds as in Code Example 12.

Using these functions, we can now calculate the bootstrap standard error, bias and 95% confidence interval for $\hat{\theta}$, setting $B = 10^4$:

[6] Named after Vilfredo Pareto (1848–1923), the highly influential economist, political scientist, and proto-Fascist.

[7] Most of the distribution, for ordinary people, roughly conforms to a log-normal.

[8] For the data source and a fuller analysis, see Clauset *et al.* (2009).




```
plot.survival.loglog(wealth, xlab = "Net worth (dollars)", ylab = "Fraction of top 400 above that wo
rug(wealth, side = 1, col = "grey")
curve((n.tail/400) * ppareto(x, threshold = x0, exponent = wealth.pareto$exponent,
    lower.tail = FALSE), add = TRUE, lty = "dashed", from = x0, to = 2 * max(wealth))
```

Figure 6.2 Upper cumulative distribution function (or "survival function") of net worth for the 400 richest individuals in the US (2000 data). The solid line shows the fraction of the 400 individuals whose net worth W equaled or exceeded a given value w, $\Pr(W \geq w)$. (Note the logarithmic scale for both axes.) The dashed line is a maximum-likelihood estimate of the Pareto distribution, taking $x_0 = \$9 \times 10^8$. (This threshold was picked using the method of Clauset *et al.* 2009.) Since there are 302 individuals at or above the threshold, the cumulative distribution function of the Pareto has to be reduced by a factor of (302/400).

```
sim.wealth <- function() {
    rpareto(n = n.tail, threshold = wealth.pareto$xmin, exponent = wealth.pareto$exponent)
}

est.pareto <- function(data) {
    pareto.fit(data, threshold = x0)$exponent
}
```

CODE EXAMPLE 12: *Simulator and estimator for model-based bootstrapping of the Pareto distribution.*

```
pareto.se <- bootstrap.se(statistic = est.pareto, simulator = sim.wealth, B = 10000)
pareto.bias <- bootstrap.bias(statistic = est.pareto, simulator = sim.wealth, t.hat = wealth.pareto$
    B = 10000)
pareto.ci <- bootstrap.ci(statistic = est.pareto, simulator = sim.wealth, B = 10000,
    t.hat = wealth.pareto$exponent, level = 0.95)
```

This gives a standard error of ± 0.078, matching the asymptotic approximation reasonably well[9], but not needing asymptotic assumptions.

Asymptotically, the bias is known to go to zero; at this size, bootstrapping gives a bias of 0.0059, which is effectively negligible.

We can also get the confidence interval; with the same 10^4 replications, the 95% CI is $2.17, 2.48$. In theory, the confidence interval could be calculated exactly, but it involves the inverse gamma distribution (Arnold, 1983), and it is quite literally faster to write and do the bootstrap than go to look it up.

A more challenging problem is goodness-of-fit; we'll use the Kolmogorov-Smirnov statistic.[10] Code Example 13 calculates the p-value. With ten thousand bootstrap replications,

```
signif(ks.pvalue.pareto(10000, wealth, wealth.pareto$exponent, x0), 4)
## [1] 0.0131
```

Ten thousand replicates is enough that we should be able to accurately estimate probabilities of around 0.01 (since the binomial standard error will be $\sqrt{\frac{(0.01)(0.99)}{10^4}} \approx 9.9 \times 10^{-4}$); if it weren't, we might want to increase B.

[9] "In Asymptopia", the variance of the MLE should be $\frac{(\hat{\theta}-1)^2}{n}$, in this case 0.076. The intuition is that this variance depends on how sharp the maximum of the likelihood function is — if it's sharply peaked, we can find the maximum very precisely, but a broad maximum is hard to pin down. Variance is thus inversely proportional to the second derivative of the negative log-likelihood. (The minus sign is because the second derivative has to be negative at a maximum, while variance has to be positive.) For one sample, the expected second derivative of the negative log-likelihood is $(\theta - 1)^{-2}$. (This is called the **Fisher information** of the model.) Log-likelihood adds across independent samples, giving us an over-all factor of n. In the large-sample limit, the actual log-likelihood will converge on the expected log-likelihood, so this gives us the asymptotic variance. (See also §**??**.)

[10] The `pareto.R` file contains a function, `pareto.tail.ks.test`, which does a goodness-of-fit test for fitting a power-law to the tail of the distribution. That differs somewhat from what follows, because it takes into account the extra uncertainty which comes from having to estimate x_0. Here, I am pretending that an Oracle told us $x_0 = 9 \times 10^8$.

```
ks.stat.pareto <- function(x, exponent, x0) {
    x <- x[x >= x0]
    ks <- ks.test(x, ppareto, exponent = exponent, threshold = x0)
    return(ks$statistic)
}

ks.pvalue.pareto <- function(B, x, exponent, x0) {
    testhat <- ks.stat.pareto(x, exponent, x0)
    testboot <- vector(length = B)
    for (i in 1:B) {
        xboot <- rpareto(length(x), exponent = exponent, threshold = x0)
        exp.boot <- pareto.fit(xboot, threshold = x0)$exponent
        testboot[i] <- ks.stat.pareto(xboot, exp.boot, x0)
    }
    p <- (sum(testboot >= testhat) + 1)/(B + 1)
    return(p)
}
```

Code Example 13: *Calculating a p-value for the Pareto distribution, using the Kolmogorov-Smirnov test and adjusting for the way estimating the scaling exponent moves the fitted distribution closer to the data.*

Simply plugging in to the standard formulas, and thereby ignoring the effects of estimating the scaling exponent, gives a p-value of 0.171, which is not outstanding but not awful either. Properly accounting for the flexibility of the model, however, the discrepancy between what it predicts and what the data shows is so large that it would take a big (one-in-a-hundred) coincidence to produce it. We have, therefore, detected that the Pareto distribution makes systematic errors for this data, but we don't know much about what they are. In Chapter F, we'll look at techniques which can begin to tell us something about *how* it fails.

```
resample <- function(x) {
    sample(x, size = length(x), replace = TRUE)
}

resample.data.frame <- function(data) {
    sample.rows <- resample(1:nrow(data))
    return(data[sample.rows, ])
}
```

CODE EXAMPLE 14: *A utility function to resample from a vector, and another which resamples from a data frame. Can you write a single function which determines whether its argument is a vector or a data frame, and does the right thing in each case/*

6.3 Bootstrapping by Resampling

The bootstrap approximates the sampling distribution, with three sources of approximation error. First, **simulation error**: using finitely many replications to stand for the full sampling distribution. Clever simulation design can shrink this, but brute force — just using enough replicates — can also make it arbitrarily small. Second, **statistical error**: the sampling distribution of the bootstrap re-estimates under our estimated model is not exactly the same as the sampling distribution of estimates under the true data-generating process. The sampling distribution changes with the parameters, and our initial estimate is not completely accurate. But it often turns out that distribution of estimates *around* the truth is more nearly invariant than the distribution of estimates themselves, so subtracting the initial estimate from the bootstrapped values helps reduce the statistical error; there are many subtler tricks to the same end. Third, **specification error**: the data source doesn't exactly follow our model at all. Simulating the model then never quite matches the actual sampling distribution.

Efron had a second brilliant idea, which is to address specification error by replacing simulation from the model with re-sampling from the data. After all, our initial collection of data gives us a lot of information about the relative probabilities of different values. In a sense the empirical distribution is the least prejudiced estimate possible of the underlying distribution — anything else imposes biases or pre-conceptions, possibly accurate but also potentially misleading[11]. Lots of quantities can be estimated directly from the empirical distribution, without the mediation of a model. Efron's **resampling bootstrap** (a.k.a. the **non-parametric bootstrap**) treats the original data set as a complete population and draws a new, simulated sample from it, picking each observation with equal probability (allowing repeated values) and then re-running the estimation (Figure 6.3, Code Example 14). In fact, this is usually what people mean when they talk about "the bootstrap" without any modifier.

Everything we did with model-based bootstrapping can also be done with re-sampling bootstrapping — the only thing that's changing is the distribution the surrogate data is coming from.

[11] See §14.6 in Chapter 14.

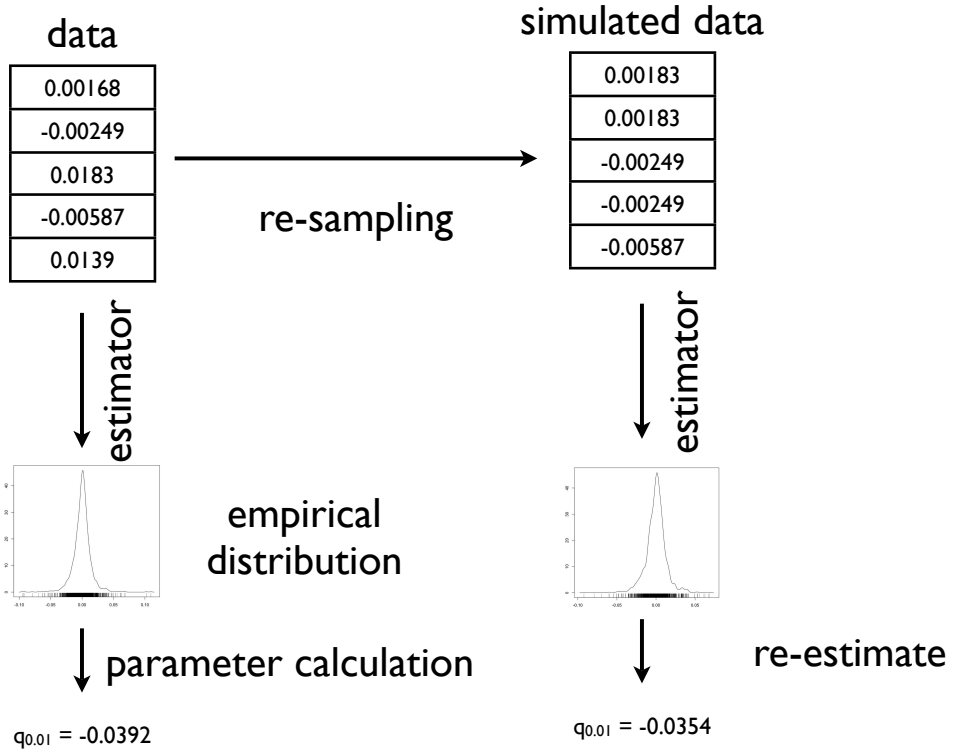


Figure 6.3 Schematic for the resampling bootstrapping. New data is simulated by re-sampling from the original data (with replacement), and functionals are calculated either directly from the empirical distribution, or by estimating a model on this surrogate data.

The resampling bootstrap should remind you of k-fold cross-validation. The analog of leave-one-out CV is a procedure called the **jack-knife**, where we repeat the estimate n times on $n-1$ of the data points, holding each one out in turn. It's historically important (it dates back to the 1940s), but generally doesn't work as well as resampling.

An important variant is the **smoothed bootstrap**, where we re-sample the data points and then perturb each by a small amount of noise, generally Gaussian[12].

Back to the Pareto example

Let's see how to use re-sampling to get a 95% confidence interval for the Pareto exponent[13].

[12] We will see in Chapter 14 that this corresponds to sampling from a kernel density estimate.

[13] Even if the Pareto model is wrong, the estimator of the exponent will converge on the value which gives, in a certain sense, the best approximation to the true distribution from among all power laws. Econometricians call such parameter values the **pseudo-truth**; we are getting a confidence interval for the pseudo-truth. In this case, the pseudo-true scaling exponent can still be a useful way of

```
wealth.resample <- function() {
    resample(wealth[wealth >= x0])
}
pareto.CI.resamp <- bootstrap.ci(statistic = est.pareto, simulator = wealth.resample,
    t.hat = wealth.pareto$exponent, level = 0.95, B = 10000)
```

The interval is 2.17, 2.48; this is very close to the interval we got from the model-based bootstrap, which should actually reassure us about the latter's validity.

6.3.1 Model-Based vs. Resampling Bootstraps

When we have a properly specified model, simulating from the model gives more accurate results (at the same n) than does re-sampling the empirical distribution — parametric estimates of the distribution converge faster than the empirical distribution does. If on the other hand the model is mis-specified, then it is rapidly converging to the *wrong* distribution. This is of course just another bias-variance trade-off, like those we've seen in regression.

Since I am suspicious of most parametric modeling assumptions, I prefer re-sampling, when I can figure out how to do it, or at least until I have convinced myself that a parametric model is a good approximation to reality.

summarizing *how* heavy tailed the income distribution is, despite the fact that the power law makes systematic errors.

6.4 Bootstrapping Regression Models

Let's recap what we're doing estimating regression models. We want to learn the regression function $\mu(x) = \mathbb{E}[Y|X = x]$. We estimate the model on a set of predictor-response pairs, $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, resulting in an estimated curve (or surface) $\widehat{\mu}(x)$, fitted values $\widehat{\mu}_i = \widehat{\mu}(x_i)$, and residuals, $\epsilon_i = y_i - \widehat{\mu}_i$. For any such model, we have a choice of several ways of bootstrapping, in decreasing order of reliance on the model.

- Simulate new X values from the model's distribution of X, and then draw Y from the specified conditional distribution $Y|X$.
- Hold the x fixed, but draw $Y|X$ from the specified distribution.
- Hold the x fixed, but make Y equal to $\widehat{\mu}(x)$ plus a randomly re-sampled ϵ_j.
- Re-sample (x, y) pairs.

The first case is pure model-based bootstrapping. (So is the second, sometimes, when the regression model is agnostic about X.) The last case is just re-sampling from the joint distribution of (X, Y). The next-to-last case is called **re-sampling the residuals** or **re-sampling the errors**. When we do that, we rely on the regression model to get the conditional expectation function right, but we don't count on it getting the distribution of the noise around the expectations.

The specific procedure of re-sampling the residuals is to re-sample the ϵ_i, with replacement, to get $\tilde{\epsilon}_1, \tilde{\epsilon}_2, \ldots \tilde{\epsilon}_n$, and then set $\tilde{x}_i = x_i$, $\tilde{y}_i = \widehat{\mu}(\tilde{x}_i) + \tilde{\epsilon}_i$. This surrogate data set is then re-analyzed like new data.

6.4.1 Re-sampling Points: Parametric Model Example

A classic data set contains the time between 299 eruptions of the Old Faithful geyser in Yellowstone, and the length of the subsequent eruptions; these variables are called `waiting` and `duration`. (We saw this data set already in §5.4.2.1, and will see it again in §10.3.2.) We'll look at the linear regression of `waiting` on `duration`. We'll re-sample (`duration`, `waiting`) pairs, and would like confidence intervals for the regression coefficients. This is a confidence interval for the coefficients of *the best linear predictor*, a functional of the distribution, which, as we saw in Chapters 1 and 2, exists no matter how nonlinear the process really is. It's only a confidence interval for the *true regression parameters* if the real regression function is linear.

Before anything else, look at the model:

```
library(MASS)
data(geyser)
geyser.lm <- lm(waiting ~ duration, data = geyser)
```

	Estimate	Std. Error	t value	Pr(¿—t—)
(Intercept)	99.3	1.960	50.7	0
duration	-7.8	0.537	-14.5	0

The first step in bootstrapping this is to build our simulator, which just means sampling rows from the data frame:

```
resample.geyser <- function() {
    resample.data.frame(geyser)
}
```

We can check this by running `summary(geyser.resample())`, and seeing that it gives about the same quartiles and mean for both variables as `summary(geyser)`[14], but that the former gives different numbers each time it's run.

Next, we define the estimator:

```
est.geyser.lm <- function(data) {
    fit <- lm(waiting ~ duration, data = data)
    return(coefficients(fit))
}
```

We can check that this function works by seeing that `coefficients(geyser.lm)` matches `est.geyser.lm(geyser)`, but that `est.geyser.lm(resample.geyser()` is different every time we run it.

Put the pieces together:

```
geyser.lm.ci <- bootstrap.ci(statistic=est.geyser.lm,
                             simulator=resample.geyser,
                             level=0.95,
                             t.hat=coefficients(geyser.lm),
                             B=1e4)
```

	lower	upper
(Intercept)	96.50	102.00
duration	-8.69	-6.91

Notice that we do not have to assume homoskedastic Gaussian noise — fortunately, because that's a very bad assumption here[15].

[14] The minimum and maximum won't match up well — why not?

[15] We have calculated 95% confidence intervals for the intercept β_0 and the slope β_1 separately. These intervals cover their coefficients all but 5% of the time. Taken together, they give us a rectangle in (β_0, β_1) space, but the coverage probability of *this* rectangle could be anywhere from 95% all the way down to 90%. To get a confidence *region* which simultaneously covers both coefficients 95% of the time, we have two big options. One is to stick to a box-shaped region and just increase the confidence level on each coordinate (to 97.5%). The other is to define some suitable metric of how far apart coefficient vectors are (e.g., ordinary Euclidean distance), find the 95% percentile of the distribution of this metric, and trace the appropriate contour around $\hat{\beta}_0, \hat{\beta}_1$.

```
main.curve <- npr.geyser(geyser)

# We already defined this in a previous example, but it doesn't hurt
resample.geyser <- function() { resample.data.frame(geyser) }

geyser.resampled.curves <- rboot(statistic=npr.geyser,
                                 simulator=resample.geyser,
                                 B=800)
```

CODE EXAMPLE 15: *Generating multiple kernel-regression curves for the geyser data, by resampling that data frame and re-estimating the model on each simulation.* `geyser.resampled.curves` *stores the predictions of those 800 models, evaluated at a* common *set of values for the predictor variable. The vector* `main.curve`*, which we'll use presently to get confidence intervals, stores predictions of the model fit to the whole data, evaluated at that same set of points.*

6.4.2 Re-sampling Points: Non-parametric Model Example

Nothing in the logic of re-sampling data points for regression requires us to use a parametric model. Here we'll provide 95% confidence bounds for the kernel smoothing of the geyser data. Since the functional is a whole curve, the confidence set is often called a **confidence band**.

We use the same simulator, but start with a different regression curve, and need a different estimator.

```
evaluation.points <- data.frame(duration = seq(from = 0.8, to = 5.5, length.out = 200))

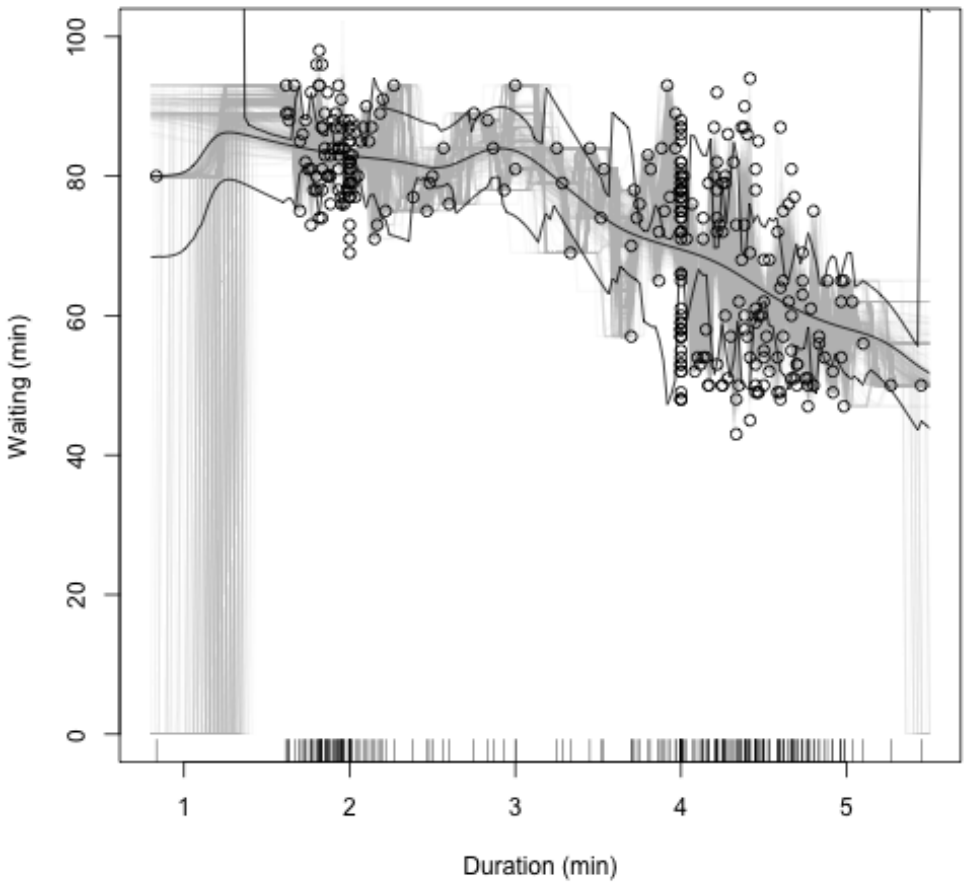
library(np)

npr.geyser <- function(data, tol = 0.1, ftol = 0.1, plot.df = evaluation.points) {
    bw <- npregbw(waiting ~ duration, data = data, tol = tol, ftol = ftol)
    mdl <- npreg(bw)
    return(predict(mdl, newdata = plot.df))
}
```

Now we construct pointwise 95% confidence bands for the regression curve. For this end, we don't really need to keep around the whole kernel regression object — we'll just use its predicted values on a uniform grid of points, extending slightly beyond the range of the data (Code Example 15). Observe that this will go through bandwidth selection again for each bootstrap sample. This is slow, but it is the most secure way of getting good confidence bands. Applying the bandwidth we found on the data to each re-sample would be faster, but would introduce an extra level of approximation, since we wouldn't be treating each simulation run the same as the original data.

Figure 6.4 shows the curve fit to the data, the 95% confidence limits, and (faintly) all of the bootstrapped curves. Doing the 800 bootstrap replicates took 4 minutes on my laptop[16].

[16] Specifically, I ran `system.time(geyser.resampled.curves <- rboot(statistic=npr.geyser, simulator=resample.geyser, B=800))`, which not only did the calculations and stored them in `geyser.resampled.curves`, but told me how much time it took R to do all that.




```
plot(0, type = "n", xlim = c(0.8, 5.5), ylim = c(0, 100), xlab = "Duration (min)",
    ylab = "Waiting (min)")
for (i in 1:ncol(geyser.resampled.curves)) {
    lines(evaluation.points$duration, geyser.resampled.curves[, i], lwd = 0.1, col = "grey")
}
geyser.npr.cis <- bootstrap.ci(tboots = geyser.resampled.curves, t.hat = main.curve,
    level = 0.95)
lines(evaluation.points$duration, geyser.npr.cis[, "lower"])
lines(evaluation.points$duration, geyser.npr.cis[, "upper"])
lines(evaluation.points$duration, main.curve)
rug(geyser$duration, side = 1)
points(geyser$duration, geyser$waiting)
```

Figure 6.4 Kernel regression curve for Old Faithful (central black line), with 95% confidence bands (other black lines), the 800 bootstrapped curves (thin, grey lines), and the data points. Notice that the confidence bands get wider where there is less data. *Caution:* doing the bootstrap took 4 minutes to run on my computer.

```
resample.residuals.penn <- function() {
    new.frame <- penn
    new.growths <- fitted(penn.lm) + resample(residuals(penn.lm))
    new.frame$gdp.growth <- new.growths
    return(new.frame)
}

penn.estimator <- function(data) {
    mdl <- lm(penn.formula, data = data)
    return(coefficients(mdl))
}

penn.lm.cis <- bootstrap.ci(statistic = penn.estimator, simulator = resample.residuals.penn,
    B = 10000, t.hat = coefficients(penn.lm), level = 0.95)
```

Code Example 16: *Re-sampling the residuals to get confidence intervals in a linear model.*

6.4.3 Re-sampling Residuals: Example

As an example of re-sampling the residuals, rather than data points, let's take a linear regression, based on the data-analysis assignment in §11. We will regress `gdp.growth` on `log(gdp)`, `pop.growth`, `invest` and `trade`:

```
penn <- read.csv("http://www.stat.cmu.edu/~cshalizi/uADA/13/hw/02/penn-select.csv")
penn.formula <- "gdp.growth ~ log(gdp) + pop.growth + invest + trade"
penn.lm <- lm(penn.formula, data = penn)
```

(Why make the formula a separate object here?) The estimated parameters are

	x
(Intercept)	5.71e-04
log(gdp)	5.07e-04
pop.growth	-1.87e-01
invest	7.15e-04
trade	3.11e-05

Code Example 16 shows the new simulator for this set-up (`resample.residuals.penn`)[17], the new estimation function (`penn.estimator`)[18], and the confidence interval calculation (`penn.lm.cis`):

[17] How would you check that this worked?
[18] How would you check that this worked?

	lower	upper
(Intercept)	-1.53e-02	1.69e-02
log(gdp)	-1.40e-03	2.41e-03
pop.growth	-3.57e-01	-1.32e-02
invest	4.95e-04	9.42e-04
trade	-2.07e-05	8.33e-05

Doing ten thousand linear regressions took 45 seconds on my computer, as opposed to 4 minutes for eight hundred kernel regressions.

6.5 Bootstrap with Dependent Data

If the data points we are looking at are vectors (or more complicated structures) with dependence between components, but each data point is independently generated from the same distribution, then dependence isn't really an issue. We re-sample vectors, or generate vectors from our model, and proceed as usual. In fact, that's what we've done so far in several cases.

If there is dependence *across* data points, things are more tricky. If our model incorporates this dependence, then we can just simulate whole data sets from it. An appropriate re-sampling method is trickier — just re-sampling individual data points destroys the dependence, so it won't do. We will revisit this question when we look at time series in Chapter 23.

6.6 Confidence Bands for Nonparametric Regression

Many of the examples in this chapter use bootstrapping to get confidence bands for nonparametric regression. It is worth mentioning that there is a subtle issue with doing so, but one which I do not think really matters, usually, for practice.

The issue is that when we do nonparametric regression, we accept some bias in our estimate of the regression function. In fact, we saw in Chapter 4 that minimizing the total MSE means accepting *matching* amounts of bias and variance. So our nonparametric estimate of μ is biased. If we simulate from it, we're simulating from something biased; if we simulate from the residuals, those residuals contain bias; and even if we do a pure resampling bootstrap, we're comparing the bootstrap replicates to a biased estimate. This means that we are really looking at sampling intervals around the biased estimate, rather than confidence intervals around μ.

The two questions this raises are (1) how much this matters, and (2) whether there is any alternative. As for the size of the bias, we know from Chapter 4 that the squared bias, in 1D, goes like $n^{-4/5}$, so the bias itself goes like $n^{-2/5}$. This does go to zero, but slowly.

[[Living with it vs. Hall and Horowitz (2013) paper, which gives $1-\alpha$ coverage at $1-\eta$ fraction of points. Essentially, construct naive bands, and then work out by how much they need to be expanded to achieve desired coverage]]

6.7 Things Bootstrapping Does Poorly

The principle behind bootstrapping is that sampling distributions under the true process should be close to sampling distributions under good estimates of the truth. If small perturbations to the data-generating process produce huge swings in the sampling distribution, bootstrapping will not work well, and may fail spectacularly. For model-based bootstrapping, this means that small changes to the underlying parameters must produce small changes to the functionals of interest.

Similarly, for resampling, it means that adding or removing a few data points must change the functionals only a little[19].

Re-sampling in particular has trouble with extreme values. Here is a simple example: Our data points X_i are IID, with $X_i \sim Unif(0, \theta_0)$, and we want to estimate θ_0. The maximum likelihood estimate $\hat{\theta}$ is just the sample maximum of the x_i. We'll use resampling to get a confidence interval for this, as above — but I will fix the true $\theta_0 = 1$, and see how often the 95% confidence interval covers the truth.

```
max.boot.ci <- function(x, B) {
    max.boot <- replicate(B, max(resample(x)))
    return(2 * max(x) - quantile(max.boot, c(0.975, 0.025)))
}
boot.cis <- replicate(1000, max.boot.ci(x = runif(100), B = 1000))
(true.coverage <- mean((1 >= boot.cis[1, ]) & (1 <= boot.cis[2, ])))
## [1] 0.877
```

That is, the actual coverage probability is not 95% but about 88%.

If you suspect that your use of the bootstrap may be setting yourself up for a similar epic fail, your two options are (1) learn some of the theory of the bootstrap from the references in the "Further Reading" section below, or (2) set up a simulation experiment like this one.

[19] More generally, moving from one distribution function f to another $(1-\epsilon)f + \epsilon g$ mustn't change the functional very much when ϵ is small, no matter in what "direction" g we perturb it. Making this idea precise calls for some fairly deep mathematics, about differential calculus on spaces of functions (see, e.g., van der Vaart 1998, ch. 20).

6.8 Which Bootstrap When?

This chapter has introduced a bunch of different bootstraps, and before it closes it's worth reviewing the general principles, and some of the considerations which go into choosing among them in a particular problem.

When we bootstrap, we try to approximate the sampling distribution of some statistic (mean, median, correlation coefficient, regression coefficients, smoothing curve, difference in MSEs...) by running simulations, and calculating the statistic on the simulation. We've seen three major ways of doing this:

- The model-based bootstrap: we estimate the model, and then simulate from x the estimated model;
- Resampling residuals: we estimate the model, and then simulate by resampling residuals to that estimate and adding them back to the fitted values;
- Resampling cases or whole data points: we ignore the estimated model completely in our simulation, and just re-sample whole rows from the data frame.

Which kind of bootstrap is appropriate depends on how much trust we have in our model.

The model-based bootstrap trusts the model to be completely correct for *some* parameter value. In, e.g., regression, it trusts that we have the right shape for the regression function *and* that we have the right distribution for the noise. When we trust our model this much, we could in principle work out sampling distributions analytically; the model-based bootstrap replaces hard math with simulation.

Resampling residuals doesn't trust the model as much. In regression problems, it assumes that the model gets the *shape* of the regression function right, and that the noise around the regression function is independent of the predictor variables, but doesn't make any further assumption about how the fluctuations are distributed. It is therefore more secure than model-based bootstrap.[20]

Finally, resampling cases assumes nothing at all about either the shape of the regression function or the distribution of the noise, it just assumes that each data point (row in the data frame) is an independent observation. Because it assumes so little, and doesn't depend on any particular model being correct, it is very safe.

The reason we do not always use the safest bootstrap, which is resampling cases, is that there is, as usual, a bias-variance trade-off. Generally speaking, if we compare three sets of bootstrap confidence intervals on the same data for the same statistic, the model-based bootstrap will give the narrowest intervals, followed by resampling residuals, and resampling cases will give the loosest bounds. If the model really *is* correct about the shape of the curve, we can get more precise results, without any loss of accuracy, by resampling residuals rather than

[20] You could also imagine simulations where we presume that the noise takes a very particular form (e.g., a t-distribution with 10 degrees of freedom), but are agnostic about the shape of the regression function, and learn that non-parametrically. It's harder to think of situations where this is really plausible, however, except *maybe* Gaussian noise arising from central-limit-theorem considerations.

resampling cases. If the model is also correct about the distribution of noise, we can do even better with a model-based bootstrap.

To sum up: resampling cases is safer than resampling residuals, but gives wider, weaker bounds. If you have good reason to trust a model's guess at the shape of the regression function, then resampling residuals is preferable. If you don't, or it's not a regression problem so there are no residuals, then you prefer to resample cases. The model-based bootstrap works best when the over-all model is correct, and we're just uncertain about the exact parameter values we need.

6.9 Further Reading

Davison and Hinkley (1997) is both a good textbook, and the reference I consult most often. Efron and Tibshirani (1993), while also very good, is more theoretical. Canty *et al.* (2006) has useful advice for serious applications.

All the bootstraps discussed in this chapter presume IID observations. For bootstraps for time series, see §23.5.

Software

For professional purposes, I strongly recommend using the R package `boot` (Canty and Ripley, 2013), based on Davison and Hinkley (1997). I deliberately do *not* use it in this chapter, or later in the book, for pedagogical reasons; I have found that forcing students to write their own bootstrapping code helps build character, or at least understanding.

The bootstrap vs. robust standard errors

For linear regression coefficients, econometricians have developed a variety of "robust" standard errors which are valid under weaker conditions than the usual assumptions. Buja *et al.* (2014) shows their equivalence to resampling cases. (See also King and Roberts 2015.)

Historical notes

The original paper on the bootstrap, Efron (1979), is extremely clear, and for the most part presented in the simplest possible terms; it's worth reading. His later small book (Efron, 1982), while often cited, is not in my opinion so useful nowadays[21].

As the title of that last reference suggests, the bootstrap is in some ways a successor to an older method, apparently dating back to the 1940s if not before, called the "jackknife", in which each data point is successively held back and the estimate is re-calculated; the variance of these re-estimates, appropriately scaled, is then taken as the variance of estimation, and similarly for the bias[22].

[21] It seems to have done a good job of explaining things to people who were already professional statisticians in 1982.

[22] A "jackknife" is a knife with a blade which folds into the handle; think of the held-back data point as the folded-away blade.

The jackknife is appealing in its simplicity, but is only valid under much stronger conditions than the bootstrap.

Exercises

6.1 Show that x_0 is the mode of the Pareto distribution.

6.2 Derive the maximum likelihood estimator for the Pareto distribution (Eq. 6.15) from the density (Eq. 6.14).

6.3 Show that the MLE of the Pareto distribution is consistent.

1. Using the law of large numbers, show that $\hat{\theta}$ (Eq. 6.15) converges to a limit which depends on $\mathbb{E}[\log X/x_0]$.
2. Find an expression for $\mathbb{E}[\log X/x_0]$ in terms of θ and from the density (Eq. 6.14). *Hint:* Write $\mathbb{E}[\log X/x_0]$ as an integral, change the variable of integration from x to $z = \log(x/x_0)$, and remember that the mean of an exponential random variable with rate λ is $1/\lambda$.

6.4 Find confidence bands for the linear regression model of §6.4.1 using

1. The usual Gaussian assumptions (*hint*: try the `intervals="confidence"` option to `predict`);
2. Resampling of residuals; and
3. Resampling of cases.

6.5 (Computational) Writing new functions to simulate every particular linear model is somewhat tedious.

1. Write a function which takes, as inputs, an `lm` model and a data frame, and returns a new data frame where the response variable is replaced by the model's predictions plus Gaussian noise, but all other columns are left alone.
2. Write a function which takes, as inputs, an `lm` model and a data frame, and returns a new data frame where the response variable is replaced by the model's predictions plus resampled residuals.
3. Will your functions work with `npreg` models, as well as `lm` models? If not, what do you have to modify?

Hint: See Code Example 3 in Chapter 3 for some R tricks to extract the name of the response variable from the estimated model.

7

Splines

7.1 Smoothing by Penalizing Curve Flexibility

Let's go back to the problem of smoothing one-dimensional data. We have data points $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$, and we want to find a good approximation $\widehat{\mu}$ to the true conditional expectation or regression function μ. Previously, we controlled how smooth we made $\widehat{\mu}$ indirectly, through the bandwidth of our kernels. But why not be more direct, and control smoothness itself?

A natural way to do this is to minimize the **spline objective function**

$$\mathcal{L}(m, \lambda) \equiv \frac{1}{n}\sum_{i=1}^{n}(y_i - m(x_i))^2 + \lambda \int (m''(x))^2 dx \tag{7.1}$$

The first term here is just the mean squared error of using the curve $m(x)$ to predict y. We know and like this; it is an old friend.

The *second* term, however, is something new for us. m'' is the second derivative of m with respect to x — it would be zero if m were linear, so this measures the **curvature** of m at x. The sign of $m''(x)$ says whether the curvature at x is concave or convex, but we don't care about that so we square it. We then integrate this over all x to say how curved m is, on average. Finally, we multiply by λ and add that to the MSE. This is adding a **penalty** to the MSE criterion — given two functions with the same MSE, we prefer the one with less average curvature. We will accept changes in m that increase the MSE by 1 unit if they also reduce the average curvature by at least λ.

The curve or function which solves this minimization problem,

$$\widehat{\mu}_\lambda = \operatorname*{argmin}_m \mathcal{L}(m, \lambda) \tag{7.2}$$

is called a **smoothing spline**, or **spline curve**. The name "spline" comes from a simple tool used by craftsmen to draw smooth curves, which was a thin strip of a flexible material like a soft wood; you pin it in place at particular points, called **knots**, and let it bend between them. (When the gas company dug up my front yard and my neighbor's driveway, the contractors who put everything back used a plywood board to give a smooth, curved edge to the new driveway. That board was a spline, and the knots were pairs of metal stakes on either side of the board. Figure 7.1 shows the spline after concrete was poured on one side of it.) Bending the spline takes energy — the stiffer the material, the more energy has to go into bending it through the same shape, and so the material makes a straighter curve

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/



Figure 7.1 A wooden spline used to create a smooth, curved border for a paved area (Shadyside, Pittsburgh, October 2014).

between given points. For smoothing splines, using a stiffer material corresponds to increasing λ.

It is possible to show (§7.6 below) that all solutions to Eq. 7.1, no matter what the data might be, are piecewise cubic polynomials which are continuous and have continuous first and second derivatives — i.e., not only is $\widehat{\mu}$ continuous, so are $\widehat{\mu}'$ and $\widehat{\mu}''$. The boundaries between the pieces sit at the original data points. By analogy with the craftman's spline, the boundary points are called the **knots** of the smoothing spline. The function is continuous beyond the largest and smallest data points, but it is always linear in those regions.[1]

I will also assert, without proof, that, with enough pieces, such piecewise cubic polynomials can approximate any well-behaved function arbitrarily closely. Finally, smoothing splines are linear smoothers, in the sense of Chapter 1: predicted values are linear combinations of the training-set response values y_i — see Eq. 7.21 below.

7.1.1 The Meaning of the Splines

Look back to the optimization problem. As $\lambda \to \infty$, any curvature at all becomes infinitely costly, and only linear functions are allowed. But we know how to minimize mean squared error with linear functions, that's OLS. So we understand that limit.

On the other hand, as $\lambda \to 0$, we decide that we don't care about curvature. In that case, we can always come up with a function which just interpolates between the data points, an **interpolation spline** passing exactly through each point. More specifically, of the infinitely many functions which interpolate between those points, we pick the one with the minimum average curvature.

At intermediate values of λ, $\widehat{\mu}_\lambda$ becomes a function which compromises between

[1] Can you explain why it is linear outside the data range, in terms of the optimization problem?

having low curvature, and bending to approach all the data points closely (on average). The larger we make λ, the more curvature is penalized. There is a bias-variance trade-off here. As λ grows, the spline becomes less sensitive to the data, with lower variance to its predictions but more bias. As λ shrinks, so does bias, but variance grows. For consistency, we want to let $\lambda \to 0$ as $n \to \infty$, just as, with kernel smoothing, we let the bandwidth $h \to 0$ while $n \to \infty$.

We can also think of the smoothing spline as the function which minimizes the mean squared error, subject to a constraint on the average curvature. This turns on a general corresponds between penalized optimization and optimization under constraints, which is explored in Appendix D.3. The short version is that each level of λ corresponds to imposing a cap on how much curvature the function is allowed to have, on average, and the spline we fit with that λ is the MSE-minimizing curve subject to that constraint.[2] As we get more data, we have more information about the true regression function and can relax the constraint (let λ shrink) without losing reliable estimation.

It will not surprise you to learn that we select λ by cross-validation. Ordinary k-fold CV is entirely possible, but leave-one-out CV works quite well for splines. In fact, the default in most spline software is either leave-one-out CV, or the even faster approximation called "generalized cross-validation" or GCV (see §3.4.3).

7.2 Computational Example: Splines for Stock Returns

The default R function for fitting a smoothing spline is `smooth.spline`:

```
smooth.spline(x, y, cv = FALSE)
```

where `x` should be a vector of values for input variable, `y` is a vector of values for the response (in the same order), and the switch `cv` controls whether to pick λ by generalized cross-validation (the default) or by leave-one-out cross-validation. The object which `smooth.spline` returns has an `$x` component, *re-arranged in increasing order*, a `$y` component of fitted values, a `$yin` component of original values, etc. See `help(smooth.spline)` for more.

As a concrete illustration, Figure 7.2 looks at the daily logarithmic returns[3] of the S&P 500 stock index, on 5542 consecutive trading days, from 9 February 1993 to 9 February 2015[4].

[2] The slightly longer version: Consider minimizing the MSE (not the penalized MSE), but only over functions m where $\int (m''(x))^2 dx$ is at most some maximum level C. λ would then be the Lagrange multiplier enforcing the constraint. The constrained but unpenalized optimization is equivalent to the penalized but unconstrained one. In economics, λ would be called the "shadow price" of average curvature in units of MSE, the rate at which we'd be willing to pay to have the constraint level C marginally increased.

[3] For a financial asset whose price on day t is p_t and which pays a dividend on that day of d_t, the log-returns on t are $\log(p_t + d_t)/p_{t-1}$. Financiers and other professional gamblers care more about the log returns than about the price change, $p_t - p_{t-1}$, because the log returns give the rate of profit (or loss) on investment. We are using a price series which is adjusted to incorporate dividend (and related) payments.

[4] This uses the handy `pdfetch` library, which downloads data from such public domain sources as the Federal Reserve, Yahoo Finance, etc.

```
require(pdfetch)
sp <- pdfetch_YAHOO("SPY", fields = "adjclose", from = as.Date("1993-02-09"), to = as.Date("2015-02-
sp <- diff(log(sp))
sp <- sp[-1]
```

We want to use the log-returns on one day to predict what they will be on the next. The horizontal axis in the figure shows the log-returns for each of 2527 days t, and the vertical axis shows the corresponding log-return for the succeeding day $t+1$. A linear model fitted to this data displays a slope of -0.0642 (grey line in the figure). Fitting a smoothing spline with cross-validation selects $\lambda = 0.0127$, and the black curve:

```
sp.today <- head(sp, -1)
sp.tomorrow <- tail(sp, -1)
coefficients(lm(sp.tomorrow ~ sp.today))
##   (Intercept)      sp.today
##  0.0003716842 -0.0640909118
sp.spline <- smooth.spline(x = sp.today, y = sp.tomorrow, cv = TRUE)
sp.spline
## Call:
## smooth.spline(x = sp.today, y = sp.tomorrow, cv = TRUE)
##
## Smoothing Parameter  spar= 1.346152  lambda= 0.01298188 (11 iterations)
## Equivalent Degrees of Freedom (Df): 5.857222
## Penalized Criterion (RSS): 0.7807542
## PRESS(l.o.o. CV): 0.0001428134
sp.spline$lambda
## [1] 0.01298188
```

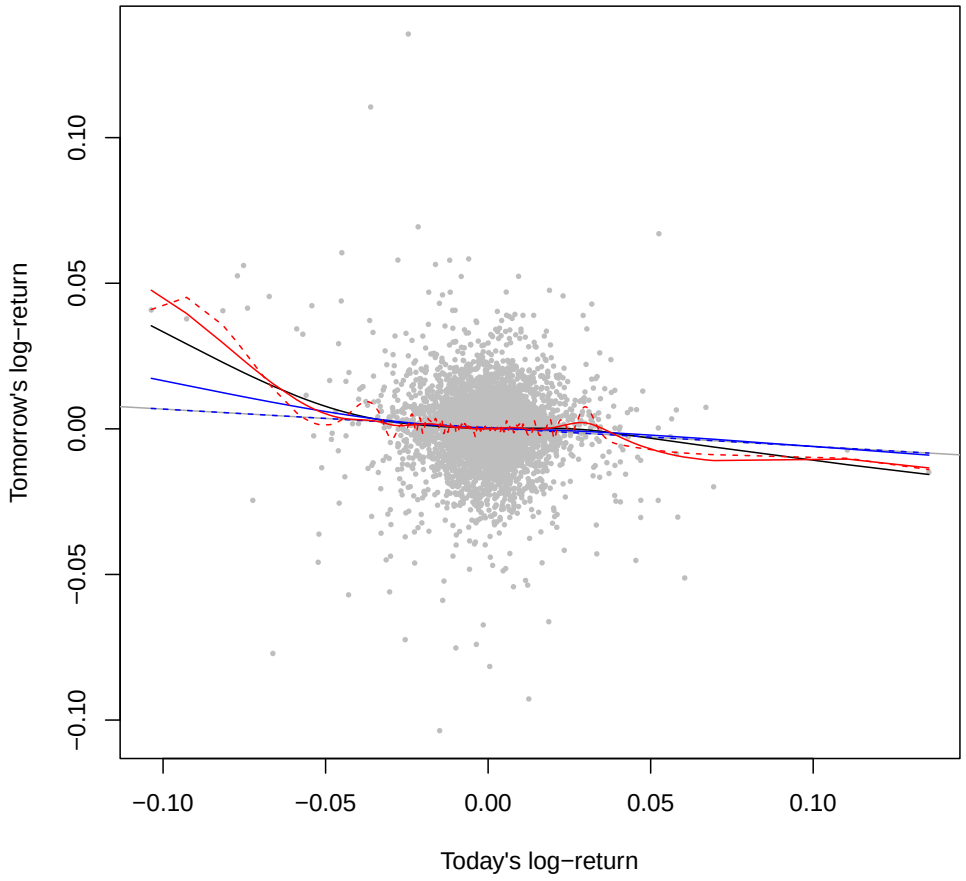
(PRESS is the "prediction sum of squares", i.e., the sum of the squared leave-one-out prediction errors.) This is the curve shown in black in the figure. The blue curves are for large values of λ, and clearly approach the linear regression; the red curves are for smaller values of λ.

The spline can also be used for prediction. For instance, if we want to know what the return to expect following a day when the log return was $+0.01$, we do

```
predict(sp.spline, x = 0.01)
## $x
## [1] 0.01
##
## $y
## [1] 0.0001949144
```

R Syntax Note:

The syntax for `predict` with `smooth.spline` spline differs slightly from the syntax for predict with `lm` or `np`. The latter two want a `newdata` argument, which should be a data-frame with column names matching those in the formula used to fit the model. The `predict` function for `smooth.spline`, though, just wants a vector called `x`. Also, while `predict` for `lm` or `np` returns a vector of predictions, `predict` for `smooth.spline` returns a list with an `x` component (in increasing order) and a `y` component, which is the sort of thing that can be put directly into `points` or `lines` for plotting.




```
plot(as.vector(sp.today), as.vector(sp.tomorrow), xlab = "Today's log-return", ylab = "Tomorrow's lo
    pch = 16, cex = 0.5, col = "grey")
abline(lm(sp.tomorrow ~ sp.today), col = "darkgrey")
sp.spline <- smooth.spline(x = sp.today, y = sp.tomorrow, cv = TRUE)
lines(sp.spline)
lines(smooth.spline(sp.today, sp.tomorrow, spar = 1.5), col = "blue")
lines(smooth.spline(sp.today, sp.tomorrow, spar = 2), col = "blue", lty = 2)
lines(smooth.spline(sp.today, sp.tomorrow, spar = 1.1), col = "red")
lines(smooth.spline(sp.today, sp.tomorrow, spar = 0.5), col = "red", lty = 2)
```

Figure 7.2 The S& P 500 log-returns data (grey dots), with the OLS linear regression (dark grey line), the spline selected by cross-validation (solid black, $\lambda = 0.0127$), some more smoothed splines (blue, $\lambda = 0.178$ and 727) and some less smooth splines (red, $\lambda = 2.88 \times 10^{-4}$ and 1.06×10^{-8}). Incoveniently, `smooth.spline` does not let us control λ directly, but rather a somewhat complicated but basically exponential transformation of it called `spar`. (See `help(smooth.spline)` for the gory details.) The equivalent λ can be extracted from the return value, e.g., `smooth.spline(sp.today,sp.tomorrow,spar=2)$lambda`.

7.2.1 Confidence Bands for Splines

Continuing the example, the smoothing spline selected by cross-validation has a negative slope everywhere, like the regression line, but it's asymmetric — the slope is more negative to the left, and then levels off towards the regression line. (See Figure 7.2 again.) Is this real, or might the asymmetry be a sampling artifact?

We'll investigate by finding confidence bands for the spline, much as we did for kernel regression in Chapter 6 and Problem Set 24, problem 5. Again, we need to bootstrap, and we can do it either by resampling the residuals or resampling whole data points. Let's take the latter approach, which assumes less about the data. We'll need a simulator:

```
sp.frame <- data.frame(today = sp.today, tomorrow = sp.tomorrow)

sp.resampler <- function() {
    n <- nrow(sp.frame)
    resample.rows <- sample(1:n, size = n, replace = TRUE)
    return(sp.frame[resample.rows, ])
}
```

This treats the points in the scatterplot as a complete population, and then draws a sample from them, with replacement, just as large as the original[5]. We'll also need an estimator. What we want to do is get a whole bunch of spline curves, one on each simulated data set. But since the values of the input variable will change from one simulation to another, to make everything comparable we'll evaluate each spline function on a fixed grid of points, that runs along the range of the data.

```
grid.300 <- seq(from = min(sp.today), to = max(sp.today), length.out = 300)

sp.spline.estimator <- function(data, eval.grid = grid.300) {
    fit <- smooth.spline(x = data[, 1], y = data[, 2], cv = TRUE)
    return(predict(fit, x = eval.grid)$y)
}
```

This sets the number of evaluation points to 300, which is large enough to give visually smooth curves, but not so large as to be computationally unwieldly.

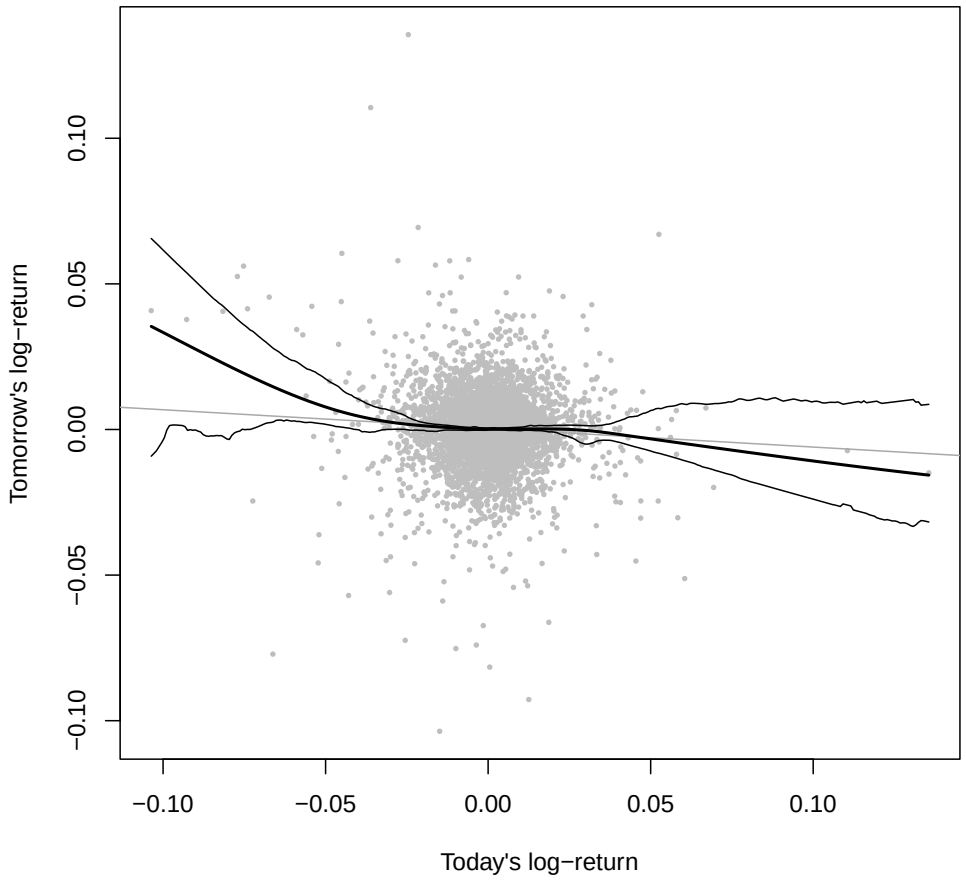
Now put these together to get confidence bands:

```
sp.spline.cis <- function(B, alpha, eval.grid = grid.300) {
    spline.main <- sp.spline.estimator(sp.frame, eval.grid = eval.grid)
    spline.boots <- replicate(B, sp.spline.estimator(sp.resampler(), eval.grid = eval.grid))
    cis.lower <- 2 * spline.main - apply(spline.boots, 1, quantile, probs = 1 - alpha/2)
    cis.upper <- 2 * spline.main - apply(spline.boots, 1, quantile, probs = alpha/2)
    return(list(main.curve = spline.main, lower.ci = cis.lower, upper.ci = cis.upper,
        x = eval.grid))
}
```

The return value here is a list which includes the original fitted curve, the lower and upper confidence limits, and the points at which all the functions were evaluated.

[5] §23.5 covers more refined ideas about bootstrapping time series.

Figure 7.3 shows the resulting 95% confidence limits, based on `B=1000` bootstrap replications. (Doing all the bootstrapping took 45 seconds on my laptop.) These are pretty clearly asymmetric in the same way as the curve fit to the whole data, but notice how wide they are, and how they get wider the further we go from the center of the distribution in either direction.




```
sp.cis <- sp.spline.cis(B = 1000, alpha = 0.05)
plot(as.vector(sp.today), as.vector(sp.tomorrow), xlab = "Today's log-return", ylab = "Tomorrow's lo
    pch = 16, cex = 0.5, col = "grey")
abline(lm(sp.tomorrow ~ sp.today), col = "darkgrey")
lines(x = sp.cis$x, y = sp.cis$main.curve, lwd = 2)
lines(x = sp.cis$x, y = sp.cis$lower.ci)
lines(x = sp.cis$x, y = sp.cis$upper.ci)
```

Figure 7.3 Bootstrapped pointwise confidence band for the smoothing spline of the S & P 500 data, as in Figure 7.2. The 95% confidence limits around the main spline estimate are based on 1000 bootstrap re-samplings of the data points in the scatterplot.

7.3 Basis Functions and Degrees of Freedom

7.3.1 Basis Functions

Splines, I said, are piecewise cubic polynomials. To see how to fit them, let's think about how to fit a global cubic polynomial. We would define four **basis functions**,

$$B_1(x) = 1 \tag{7.3}$$
$$B_2(x) = x \tag{7.4}$$
$$B_3(x) = x^2 \tag{7.5}$$
$$B_4(x) = x^3 \tag{7.6}$$

and chose to only consider regression functions that are linear combinations of the basis functions,

$$\mu(x) = \sum_{j=1}^{4} \beta_j B_j(x) \tag{7.7}$$

Such regression functions would be linear in the *transformed* variables $B_1(x), \ldots B_4(x)$, even though it is nonlinear in x.

To estimate the coefficients of the cubic polynomial, we would apply each basis function to each data point x_i and gather the results in an $n \times 4$ matrix $\mathbf{B}$,

$$B_{ij} = B_j(x_i) \tag{7.8}$$

Then we would do OLS using the $\mathbf{B}$ matrix in place of the usual data matrix $\mathbf{x}$:

$$\hat{\beta} = (\mathbf{B}^T\mathbf{B})^{-1}\mathbf{B}^T\mathbf{y} \tag{7.9}$$

Since splines are piecewise cubics, things proceed similarly, but we need to be a little more careful in defining the basis functions. Recall that we have n values of the input variable x, $x_1, x_2, \ldots x_n$. For the rest of this section, I will assume that these are in increasing order, because it simplifies the notation. These n "knots" define $n+1$ pieces or segments: $n-1$ of them between the knots, one from $-\infty$ to x_1, and one from x_n to $+\infty$. A third-order polynomial on each segment would seem to need a constant, linear, quadratic and cubic term per segment. So the segment running from x_i to x_{i+1} would need the basis functions

$$\mathbf{1}_{(x_i,x_{i+1})}(x),\ (x-x_i)\mathbf{1}_{(x_i,x_{i+1})}(x),\ (x-x_i)^2\mathbf{1}_{(x_i,x_{i+1})}(x),\ (x-x_i)^3\mathbf{1}_{(x_i,x_{i+1})}(x) \tag{7.10}$$

where as usual the indicator function $\mathbf{1}_{(x_i,x_{i+1})}(x)$ is 1 if $x \in (x_i, x_{i+1})$ and 0 otherwise. This makes it seem like we need $4(n+1) = 4n+4$ basis functions.

However, we know from linear algebra that the number of basis vectors we need is equal to the number of dimensions of the vector space. The number of adjustable coefficients for an arbitrary piecewise cubic with $n+1$ segments is indeed $4n+4$, but splines are constrained to be smooth. The spline must be continuous, which means that at each x_i, the value of the cubic from the left, defined on (x_{i-1}, x_i), must match the value of the cubic from the right, defined on (x_i, x_{i+1}). This gives us one constraint per data point, reducing the number of

adjustable coefficients to at most $3n+4$. Since the first and second derivatives are also continuous, we are down to just $n+4$ coefficients. Finally, we know that the spline function is linear outside the range of the data, i.e., on $(-\infty, x_1)$ and on (x_n, ∞), lowering the number of coefficients to n. There are no more constraints, so we end up needing only n basis functions. And in fact, from linear algebra, any set of n piecewise cubic functions which are linearly independent[6] can be used as a basis. One common choice is

$$B_1(x) = 1 \tag{7.11}$$

$$B_2(x) = x \tag{7.12}$$

$$B_{i+2}(x) = \frac{(x-x_i)_+^3 - (x-x_n)_+^3}{x_n - x_i} - \frac{(x-x_{n-1})_+^3 - (x-x_n)_+^3}{x_n - x_{n-1}} \tag{7.13}$$

where $(a)_+ = a$ if $a > 0$, and $= 0$ otherwise. This rather unintuitive-looking basis has the nice property that the second and third derivatives of each B_j are zero outside the interval (x_1, x_n).

Now that we have our basis functions, we can once again write the spline as a weighted sum of them,

$$m(x) = \sum_{j=1}^{m} \beta_j B_j(x) \tag{7.14}$$

and put together the matrix $\mathbf{B}$ where $B_{ij} = B_j(x_i)$. We can write the spline objective function in terms of the basis functions,

$$n\mathcal{L} = (\mathbf{y} - \mathbf{B}\beta)^T(\mathbf{y} - \mathbf{B}\beta) + n\lambda\beta^T\Omega\beta \tag{7.15}$$

where the matrix Ω encodes information about the curvature of the basis functions:

$$\Omega_{jk} = \int B_j''(x) B_k''(x) dx \tag{7.16}$$

Notice that only the quadratic and cubic basis functions will make non-zero contributions to Ω. With the choice of basis above, the second derivatives are non-zero on, at most, the interval (x_1, x_n), so each of the integrals in Ω is going to be finite. This is something we (or, realistically, R) can calculate *once*, no matter what λ is. Now we can find the smoothing spline by differentiating with respect to β:

$$0 = -2\mathbf{B}^T\mathbf{y} + 2\mathbf{B}^T\mathbf{B}\hat{\beta} + 2n\lambda\Omega\hat{\beta} \tag{7.17}$$

$$\mathbf{B}^T\mathbf{y} = \left(\mathbf{B}^T\mathbf{B} + n\lambda\Omega\right)\hat{\beta} \tag{7.18}$$

$$\hat{\beta} = \left(\mathbf{B}^T\mathbf{B} + n\lambda\Omega\right)^{-1}\mathbf{B}^T\mathbf{y} \tag{7.19}$$

[6] Recall that vectors $\vec{v}_1, \vec{v}_2, \ldots \vec{v}_d$ are linearly independent when there is no way to write any one of the vectors as a weighted sum of the others. The same definition applies to functions.

Notice, incidentally, that we can now show splines are linear smoothers:

$$\widehat{\mu}(x) = \mathbf{B}\hat{\beta} \tag{7.20}$$

$$= \mathbf{B}\left(\mathbf{B}^T\mathbf{B} + n\lambda\Omega\right)^{-1}\mathbf{B}^T\mathbf{y} \tag{7.21}$$

Once again, if this were ordinary linear regression, the OLS estimate of the coefficients would be $(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$. In comparison to that, we've made two changes. First, we've substituted the basis function matrix $\mathbf{B}$ for the original matrix of independent variables, $\mathbf{x}$ — a change we'd have made already for a polynomial regression. Second, the "denominator" is not $\mathbf{x}^T\mathbf{x}$, or even $\mathbf{B}^T\mathbf{B}$, but $\mathbf{B}^T\mathbf{B}+n\lambda\Omega$. Since $\mathbf{x}^T\mathbf{x}$ is n times the covariance matrix of the independent variables, we are taking the covariance matrix of the spline basis functions and adding some extra covariance — how much depends on the shapes of the functions (through Ω) and how much smoothing we want to do (through λ). The larger we make λ, the less the actual data matters to the fit.

In addition to explaining how splines can be fit quickly (do some matrix arithmetic), this illustrates two important tricks. One, which we won't explore further here, is to turn a nonlinear regression problem into one which is linear in another set of basis functions. This is like using not just one transformation of the input variables, but a whole library of them, and letting the data decide which transformations are important. There remains the issue of selecting the basis functions, which can be quite tricky. In addition to the spline basis[7], most choices are various sorts of waves — sine and cosine waves of different frequencies, various wave-forms of limited spatial extent ("wavelets"), etc. The ideal is to chose a function basis where only a few non-zero coefficients would need to be estimated, but this requires some understanding of the data...

The other trick is that of stabilizing an unstable estimation problem by adding a penalty term. This reduces variance at the cost of introducing some bias. Exercise 7.2 explores this idea.

Effective degrees of freedom

In §1.5.3.2, we defined the number of effective degrees of freedom for a linear smoother with smoothing matrix $\mathbf{w}$ as just $\mathrm{tr}\,\mathbf{w}$. Thus, Eq. 7.21 lets us calculate the effective degrees of freedom of a spline, as $\mathrm{tr}\left(\mathbf{B}(\mathbf{B}^T\mathbf{B} + n\lambda\Omega)^{-1}\mathbf{B}^T\right)$. You should be able to convince yourself from this that increasing λ will, all else being equal, reduce the effective degrees of freedom of the fit.

7.4 Splines in Multiple Dimensions

Suppose we have *two* input variables, x and z, and a single response y. How could we do a spline fit?

[7] Or, really, bases; there are multiple sets of basis functions for the splines, just like there are multiple sets of basis vectors for the plane. Phrases like "B splines" and "P splines" refer to particular choices of spline basis functions.

One approach is to generalize the spline optimization problem so that we penalize the curvature of the spline surface (no longer a curve). The appropriate penalized least-squares objective function to minimize is

$$\mathcal{L}(m,\lambda) = \sum_{i=1}^{n}(y_i - m(x_i,z_i))^2 + \lambda \int \left[\left(\frac{\partial^2 m}{\partial x^2} \right)^2 + 2\left(\frac{\partial^2 m}{\partial x \partial z} \right)^2 + \left(\frac{\partial^2 m}{\partial z^2} \right)^2 \right] dxdz \tag{7.22}$$

The solution is called a **thin-plate spline**. This is appropriate when the two input variables x and z should be treated more or less symmetrically[8].

An alternative is use the spline basis functions from section 7.3. We write

$$m(x) = \sum_{j=1}^{M_1}\sum_{k=1}^{M_2} \beta_{jk} B_j(x) B_k(z) \tag{7.23}$$

Doing all possible multiplications of one set of numbers or functions with another is said to give their **outer product** or **tensor product**, so this is known as a **tensor product spline** or **tensor spline**. We have to chose the number of terms to include for each variable (M_1 and M_2), since using n for each would give n^2 basis functions, and fitting n^2 coefficients to n data points is asking for trouble.

7.5 Smoothing Splines versus Kernel Regression

For one input variable and one output variable, smoothing splines can basically do everything which kernel regression can do[9]. The advantages of splines are their computational speed and (once we've calculated the basis functions) simplicity, as well as the clarity of controlling curvature directly. Kernels however are easier to program (if slower to run), easier to analyze mathematically[10], and extend more straightforwardly to multiple variables, and to combinations of discrete and continuous variables.

7.6 Some of the Math Behind Splines

Above, I claimed that a solution to the optimization problem Eq. 7.1 exists, and is a continuous, piecewise-cubic polynomial, with continuous first and second derivatives, with pieces at the x_i, and linear outside the range of the x_i. I do not know of any truly elementary way of showing this, but I will sketch here how it's established, if you're interested.

Eq. 7.1 asks us to find the function which minimize the sum of the MSE and

[8] Generalizations to more than two input variables are conceptually straightforward — just keep adding up more partial derivatives — but the book-keeping gets annoying.

[9] In fact, there is a technical sense in which, for large n, splines act like a kernel regression with a specific non-Gaussian kernel, and a bandwidth which varies over the data, being smaller in high-density regions. See Simonoff (1996, §5.6.2), or, for more details, Silverman (1984).

[10] Most of the bias-variance analysis for kernel regression can be done with basic calculus, as we did in Chapter 4. The corresponding analysis for splines requires working in infinite-dimensional function spaces called "Hilbert spaces". It's a pretty theory, if you like that sort of thing.

a certain integral. Even the MSE can be brought inside the integral, using Dirac delta functions:

$$\mathcal{L} = \int \left[\lambda(m''(x))^2 + \frac{1}{n}\sum_{i=1}^{n}(y_i - m(x_i))^2 \delta(x - x_i) \right] dx \tag{7.24}$$

In what follows, without loss of generality, assume that the x_i are ordered, so $x_1 \leq x_2 \leq \ldots x_i \leq x_{i+1} \leq \ldots x_n$. With some loss of generality but a great gain in simplicity, assume none of the x_i are equal, so we can make those inequalities strict.

The subject which deals with maximizing or minimizing integrals of functions is the calculus of variations[11], and one of its basic tricks is to write the integrand as a function of x, the function, and its derivatives:

$$\mathcal{L} = \int L(x, m, m', m'')dx \tag{7.25}$$

where, in our case,

$$L = \lambda(m''(x))^2 + \frac{1}{n}\sum_{i=1}^{n}(y_i - m(x_i))^2 \delta(x - x_i) \tag{7.26}$$

This sets us up to use a general theorem of the calculus of variations, to the effect that any function $\hat{m}$ which minimizes L must also solve L's **Euler-Lagrange equation**:

$$\frac{\partial L}{\partial m} - \frac{d}{dx}\frac{\partial L}{\partial m'} + \frac{d^2}{dx^2}\frac{\partial L}{\partial m''}\bigg|_{m=\hat{m}} = 0 \tag{7.27}$$

In our case, the Euler-Lagrange equation reads

$$-\frac{2}{n}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))\delta(x - x_i) + 2\lambda\frac{d^2}{dx^2}\hat{m}''(x) = 0 \tag{7.28}$$

Remembering that $\hat{m}''(x) = d^2\hat{m}/dx^2$,

$$\frac{d^4}{dx^4}\hat{m}(x) = \frac{1}{n\lambda}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))\delta(x - x_i) \tag{7.29}$$

The right-hand side is zero at any point x other than one of the x_i, so the fourth derivative has to be zero in between the x_i. This in turn means that the function must be piecewise cubic. Now fix an x_i, and pick any two points which bracket it, but are both greater than x_{i-1} and less than x_{i+1}; call them l and u. Integrate

[11] In addition to its uses in statistics, the calculus of variations also shows up in physics ("what is the path of least action?"), control theory ("what is the cheapest route to the objective?") and stochastic processes ("what is the most probable trajectory?"). Gershenfeld (1999, ch. 4) is a good starting point.

our Euler-Lagrange equation from l to u:

$$\int_l^u \frac{d^4}{dx^4}\hat{m}(x)dx = \int_l^u \frac{1}{n\lambda}\sum_{i=1}^{n}(y_i - \hat{m}(x_i))\delta(x - x_i) \tag{7.30}$$

$$\hat{m}'''(u) - \hat{m}'''(l) = \frac{y_i - \hat{m}(x_i)}{n\lambda} \tag{7.31}$$

That is, the third derivative makes a jump when we move across x_i, though (since the fourth derivative is zero), it doesn't matter which pair of points above and below x_i we compare third derivatives at. Integrating the equation again,

$$\hat{m}''(u) - \hat{m}''(l) = (u - l)\frac{y_i - \hat{m}(x_i)}{n\lambda} \tag{7.32}$$

Letting u and l approach x_i from either side, so $u - l \to 0$, we see that $\hat{m}''$ makes no jump at x_i. Repeating this trick twice more, we conclude the same about $\hat{m}'$ and $\hat{m}$ itself. In other words, $\hat{m}$ must be continuous, with continuous first and second derivatives, and a third derivative that is constant on each (x_i, x_{i+1}) interval. Since the fourth derivative is zero on those intervals (and undefined at the x_i), the function must be a piecewise cubic, with the piece boundaries at the x_i, and continuity (up to the second derivative) across pieces.

To see that the optimal function must be linear below x_1 and above x_n, suppose that it wasn't. Clearly, though, we could reduce the curvature as much as we want in those regions, without altering the value of the function at the boundary, or even its first derivative there. This would yield a better function, i.e., one with a lower value of $\mathcal{L}$, since the MSE would be unchanged and the average curvature would be smaller. Taking this to the limit, then, the function must be linear outside the observed data range.

We have now shown[12] that the optimal function $\hat{m}$, *if it exists*, must have all the properties I claimed for it. We have not shown either that there is a solution, or that a solution is unique if it does exist. However, we can use the fact that solutions, *if* there are any, are piecewise cubics obeying continuity conditions to set up a system of equations to find their coefficients. In fact, we did so already in §7.3.1, where we saw it's a system of n independent linear equations in n unknowns. Such a thing does indeed have a unique solution, here Eq. 7.19.

7.7 Further Reading

There are good discussions of splines in Simonoff (1996, ch. 5), Hastie *et al.* (2009, ch. 5) and Wasserman (2006, §5.5). Wood (2006, ch. 4) includes a thorough practical treatment of splines as a preparation for additive models (see Chapter 8 below) and generalized additive models (see Chapters 11–12). The classic reference, by one of the inventors of splines as a useful statistical tool, is Wahba (1990); it's great if you already know what a Hilbert space is and how to navigate one.

[12] For a very weak value of "shown", admittedly.

Historical notes

The first introduction of spline smoothing in the statistical literature seems to be Whittaker (1922). (His "graduation" is more or less our "smoothing".) He begins with an "inverse probability" (we would now say "Bayesian") argument for minimizing Eq. 7.1 to find the most probable curve, based on the *a priori* hypothesis of smooth Gaussian curves observed through Gaussian error, and gives tricks for fitting splines more easily with the mathematical technology available in 1922.

The general optimization problem, and the use of the word "spline", seems to have its roots in numerical analysis in the early 1960s; those spline functions were intended as ways of smoothly interpolating between given points. The connection to statistical smoothing was made by Schoenberg (1964) (who knew about Whittaker's earlier work) and by Reinsch (1967) (who gave code). Splines were then developed as a practical tool in statistics and in applied mathematics in the 1960s and 1970s. Silverman (1985) is a still-readable and insightful summary of this work.

In econometrics, spline smoothing a time series is called the "Hodrick-Prescott filter", after two economists who re-discovered the technique in 1981, along with a fallacious argument that λ should always take a particular value (1600, as it happens), regardless of the data. See Paige and Trindade (2010) for a (polite) discussion, and demonstration of the advantages of cross-validation.

Exercises

7.1 The `smooth.spline` function lets you set the effective degrees of freedom explicitly. Write a function which chooses the number of degrees of freedom by five-fold cross-validation.

7.2 When we can't measure our predictor variables perfectly, it seems like a good idea to try to include multiple measurements for each one of them. For instance, if we were trying to predict grades in college from grades in high school, we might include the student's grade from each year separately, rather than simply averaging them. Multiple measurements of the same variable will however tend to be strongly correlated, so this means that a linear regression will be *nearly* multi-collinear. This in turn means that it will tend to have multiple, mutually-canceling large coefficients. This makes it hard to interpret the regression and hard to treat the predictions seriously. (See §2.1.1.)

One strategy for coping with this situation is to carefully select the variables one uses in the regression. Another, however, is to add a penalty for large coefficient values. For historical reasons, this second strategy is called **ridge regression**, or **Tikhonov regularization**. Specifically, while the OLS estimate is

$$\widehat{\beta}_{OLS} = \underset{\beta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i \cdot \beta)^2 \ , \tag{7.33}$$

the regularized or penalized estimate is

$$\widehat{\beta}_{RR} = \underset{\beta}{\operatorname{argmin}} \left[\frac{1}{n} \sum_{i=1}^{n} (y_i - x_i \cdot \beta)^2 \right] + \lambda \sum_{j=1}^{p} \beta_j^2 \tag{7.34}$$

1. Show that the matrix form of the ridge-regression objective function is
$$n^{-1}(\mathbf{y} - \mathbf{x}\beta)^T(\mathbf{y} - \mathbf{x}\beta) + \lambda\beta^T\beta \tag{7.35}$$
2. Show that the optimum is
$$\widehat{\beta}_{RR} = (\mathbf{x}^T\mathbf{x} + n\lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y} \tag{7.36}$$
(This is where the name "ridge regression" comes from: we take $\mathbf{x}^T\mathbf{x}$ and add a "ridge" along the diagonal of the matrix.)
3. What happens as $\lambda \to 0$? As $\lambda \to \infty$? (For the latter, it may help to think about the case of a one-dimensional X first.)
4. Let $Y = Z + \epsilon$, with $Z \sim \mathcal{U}(-1, 1)$ and $\epsilon \sim \mathcal{N}(0, 0.05)$. Generate 2000 draws from Z and Y. Now let $X_i = 0.9Z + \eta$, with $\eta \sim \mathcal{N}(0, 0.05)$, for $i \in 1 : 50$. Generate corresponding X_i values. Using the first 1000 rows of the data only, do ridge regression of Y on the X_i (not on Z), plotting the 50 coefficients as functions of λ. Explain why ridge regression is called a **shrinkage estimator**.
5. Use cross-validation with the first 1000 rows to pick the optimal value of λ. Compare the out-of-sample performance you get with this penalty to the out-of-sample performance of OLS.

For more on ridge regression, see Appendix D.3.5.

8

Additive Models

8.1 Additive Models

The **additive model** for regression is that the conditional expectation function is a sum of **partial response** functions, one for each predictor variable. Formally, when the vector $\vec{X}$ of predictor variables has p dimensions, $x_1, \ldots x_p$, the model says that

$$\mathbb{E}\left[Y|\vec{X} = \vec{x}\right] = \alpha + \sum_{j=1}^{p} f_j(x_j) \tag{8.1}$$

This includes the linear model as a special case, where $f_j(x_j) = \beta_j x_j$, but it's clearly more general, because the f_js can be arbitrary nonlinear functions. The idea is still that each input feature makes a separate contribution to the response, and these just add up (hence "partial response function"), but these contributions don't have to be strictly proportional to the inputs. We do need to add a restriction to make it identifiable; without loss of generality, say that $\mathbb{E}[Y] = \alpha$ and $\mathbb{E}[f_j(X_j)] = 0$.[1]

Additive models keep a lot of the nice properties of linear models, but are more flexible. One of the nice things about linear models is that they are fairly straightforward to interpret: if you want to know how the prediction changes as you change x_j, you just need to know β_j. The partial response function f_j plays the same role in an additive model: of course the change in prediction from changing x_j will generally depend on the level x_j had before perturbation, but since that's also true of reality that's really a feature rather than a bug. It's true that a set of plots for f_js takes more room than a table of β_js, but it's also nicer to look at, conveys more information, and imposes fewer systematic distortions on the data.

Of course, none of this would be of any use if we couldn't actually estimate these models, but we can, through a clever computational trick which is worth knowing for its own sake. The use of the trick is also something they share with linear models, so we'll start there.

[1] To see why we need to do this, imagine the simple case where $p = 2$. If we add constants c_1 to f_1 and c_2 to f_2, but subtract $c_1 + c_2$ from α, then nothing *observable* has changed about the model. This degeneracy or lack of identifiability is a little like the way collinearity keeps us from defining true slopes in linear regression. But it's less harmful than collinearity because we can fix it with this convention.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

8.2 Partial Residuals and Back-fitting

8.2.1 Back-fitting for Linear Models

The general form of a linear regression model is

$$\mathbb{E}\left[Y|\vec{X}=\vec{x}\right] = \beta_0 + \vec{\beta}\cdot\vec{x} = \sum_{j=0}^{p}\beta_j x_j \tag{8.2}$$

where x_0 is always the constant 1. (Adding this fictitious constant variable lets us handle the intercept just like any other regression coefficient.)

Suppose we don't condition on all of $\vec{X}$ but just one component of it, say X_k. What is the conditional expectation of Y?

$$\mathbb{E}\left[Y|X_k=x_k\right] = \mathbb{E}\left[\mathbb{E}\left[Y|X_1,X_2,\ldots X_k,\ldots X_p\right]|X_k=x_k\right] \tag{8.3}$$

$$= \mathbb{E}\left[\sum_{j=0}^{p}\beta_j X_j|X_k=x_k\right] \tag{8.4}$$

$$= \beta_k x_k + \mathbb{E}\left[\sum_{j\neq k}\beta_j X_j|X_k=x_k\right] \tag{8.5}$$

where the first line uses the law of total expectation[2], and the second line uses Eq. 8.2. Turned around,

$$\beta_k x_k = \mathbb{E}\left[Y|X_k=x_k\right] - \mathbb{E}\left[\sum_{j\neq k}\beta_j X_j|X_k=x_k\right] \tag{8.6}$$

$$= \mathbb{E}\left[Y - \left(\sum_{j\neq k}\beta_j X_j\right)|X_k=x_k\right] \tag{8.7}$$

The expression in the expectation is the kth **partial residual** — the (total) residual is the difference between Y and its expectation, the partial residual is the difference between Y and what we expect it to be *ignoring* the contribution from X_k. Let's introduce a symbol for this, say $Y^{(k)}$.

$$\beta_k x_k = \mathbb{E}\left[Y^{(k)}|X_k=x_k\right] \tag{8.8}$$

In words, if the over-all model is linear, then the partial residuals are linear. And notice that X_k is the only input feature appearing here — if we could somehow get hold of the partial residuals, then we can find β_k by doing a simple regression, rather than a multiple regression. Of course to get the partial residual we need to know all the other β_js...

This suggests the following estimation scheme for linear models, known as the **Gauss-Seidel algorithm**, or more commonly and transparently as **back-fitting**; the pseudo-code is in Example 17.

"You say 'vicious circle', I say 'iterative improvement'."

This is an iterative approximation algorithm. Initially, we look at how far each

[2] As you learned in baby prob., this is the fact that $\mathbb{E}\left[Y|X\right] = \mathbb{E}\left[\mathbb{E}\left[Y|X,Z\right]|X\right]$ — that we can always condition more variables, provided we then average over those extra variables when we're done.

```
Given: n × (p + 1) inputs x (0th column all 1s)
       n × 1 responses y
       small tolerance δ > 0
center y and each column of x
β̂_j ← 0 for j ∈ 1 : p
until (all |β̂_j − γ_j| ≤ δ) {
       for k ∈ 1 : p {
            y_i^(k) = y_i − Σ_{j≠k} β̂_j x_ij
            γ_k ← regression coefficient of y^(k) on x_·k
            β̂_k ← γ_k
       }
}
β̂_0 ← (n^{-1} Σ_{i=1}^n y_i) − Σ_{j=1}^p β̂_j n^{-1} Σ_{i=1}^n x_ij
Return: (β̂_0, β̂_1, ... β̂_p)
```

CODE EXAMPLE 17: *Pseudocode for back-fitting linear models. Assume we make at least one pass through the* `until` *loop. Recall from Chapter 1 that centering the data does not change the* β_j*s; this way the intercept only has to be calculated once, at the end. [[ATTN: Fix horizontal lines]]*

point is from the global mean, and do a simple regression of those deviations on the first input variable. This then gives us a better idea of what the regression surface really is, and we use the deviations from *that* surface in a simple regression on the next variable; this should catch relations between Y and X_2 that weren't already caught by regressing on X_1. We then go on to the next variable in turn. At each step, each coefficient is adjusted to fit in with what we have already guessed about the other coefficients — that's why it's called "back-fitting". It is not obvious[3] that this will ever converge, but it (generally) does, and the fixed point on which it converges is the usual least-squares estimate of β.

Back-fitting is rarely used to fit linear models these days, because with modern computers and numerical linear algebra it's faster to just calculate $(\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y}$. But the cute thing about back-fitting is that it doesn't actually rely on *linearity*.

[3] Unless, I suppose, you're Gauss.

```
Given: n × p inputs x
       n × 1 responses y
       small tolerance δ > 0
       one-dimensional smoother S
α̂ ← n^{-1} Σ_{i=1}^n y_i
f̂_j ← 0 for j ∈ 1 : p
until (all |f̂_j − g_j| ≤ δ) {
       for k ∈ 1 : p {
            y_i^{(k)} = y_i − Σ_{j≠k} f̂_j(x_ij)
            g_k ← S(y^{(k)} ∼ x_·k)
            g_k ← g_k − n^{-1} Σ_{i=1}^n g_k(x_ik)
            f̂_k ← g_k
       }
}
Return: (α̂, f̂_1, ... f̂_p)
```

CODE EXAMPLE 18: *Pseudo-code for back-fitting additive models. Notice the extra step, as compared to back-fitting linear models, which keeps each partial response function centered.*

8.2.2 Backfitting Additive Models

Defining the partial residuals by analogy with the linear case, as

$$Y^{(k)} = Y - \left(\alpha + \sum_{j \neq k} f_j(x_j)\right) \tag{8.9}$$

a little algebra along the lines of §8.2.1 shows that

$$\mathbb{E}\left[Y^{(k)} | X_k = x_k\right] = f_k(x_k) \tag{8.10}$$

If we knew how to estimate arbitrary one-dimensional regressions, we could now use back-fitting to estimate additive models. But we have spent a lot of time learning how to use smoothers to fit one-dimensional regressions! We could use nearest neighbors, or splines, or kernels, or local-linear regression, or anything else we feel like substituting here.

Our new, improved back-fitting algorithm in Example 18. Once again, while it's not obvious that this converges, it does. Also, the back-fitting procedure works well with some complications or refinements of the additive model. If we know the function form of one or another of the f_j, we can fit those parametrically (rather than with the smoother) at the appropriate points in the loop. (This would be a **semiparametric** model.) If we think that there is an interaction between x_j and x_k, rather than their making separate additive contributions for each variable, we can smooth them together; etc.

8.2.3 R Implementations

There are actually *two* packages standard packages for fitting additive models in R: `gam` and `mgcv`. Both have commands called `gam`, which fit **generalized** additive models — the generalization is to use the additive model for things like the probabilities of categorical responses, rather than the response variable itself. If that sounds obscure right now, don't worry — we'll come back to this in Chapters 11–12 after we've looked at generalized linear models. §8.4 below illustrates using one of these packages to fit an additive model.

8.3 The Curse of Dimensionality

Before illustrating how additive models work in practice, let's talk about why we'd want to use them. So far, we have looked at two extremes for regression models; additive models are somewhere in between.

On the one hand, we had linear regression, which is a parametric method (with $p+1$ parameters). Its weakness is that the true regression function μ is hardly ever linear, so even with infinite data linear regression will always make systematic mistakes in its predictions — there's always some approximation bias, bigger or smaller depending on how non-linear μ is. The strength of linear regression is that it converges very quickly as we get more data. Generally speaking,

$$MSE_{\text{linear}} = \sigma^2 + a_{\text{linear}} + O(n^{-1}) \tag{8.11}$$

where the first term is the intrinsic noise around the true regression function, the second term is the (squared) approximation bias, and the last term is the estimation variance. Notice that the rate at which the estimation variance shrinks doesn't depend on p — factors like that are all absorbed into the big O.[4] Other parametric models generally converge at the same rate.

At the other extreme, we've seen a number of completely nonparametric regression methods, such as kernel regression, local polynomials, k-nearest neighbors, etc. Here the limiting approximation bias is actually *zero*, at least for any reasonable regression function μ. The problem is that they converge more slowly, because we need to use the data not just to figure out the coefficients of a parametric model, but the sheer shape of the regression function. We saw in Chapter 4 that the mean-squared error of kernel regression in one dimension is $\sigma^2 + O(n^{-4/5})$. Splines, k-nearest-neighbors (with growing k), etc., all attain the same rate. But in p dimensions, this becomes (Wasserman, 2006, §5.12)

$$MSE_{\text{nonpara}} - \sigma^2 = O(n^{-4/(p+4)}) \tag{8.12}$$

There's no ultimate approximation bias term here. Why does the rate depend on p? Well, to hand-wave a bit, think of kernel smoothing, where $\widehat{\mu}(\vec{x})$ is an average over y_i for $\vec{x}_i$ near $\vec{x}$. In a p dimensional space, the volume within ϵ of $\vec{x}$ is $O(\epsilon^p)$, so the probability that a training point $\vec{x}_i$ falls in the averaging region around $\vec{x}$ gets exponentially smaller as p grows. Turned around, to get the same number of

[4] See Appendix A you are not familiar with "big O" notation.

training points per $\vec{x}$, we need exponentially larger sample sizes. The appearance of the 4s is a little more mysterious, but can be resolved from an error analysis of the kind we did for kernel regression in Chapter 4[5]. This slow rate isn't just a weakness of kernel smoothers, but turns out to be the best any nonparametric estimator can do.

For $p = 1$, the nonparametric rate is $O(n^{-4/5})$, which is of course slower than $O(n^{-1})$, but not all that much, and the improved bias usually more than makes up for it. But as p grows, the nonparametric rate gets slower and slower, and the fully nonparametric estimate more and more imprecise, yielding the infamous **curse of dimensionality**. For $p = 100$, say, we get a rate of $O(n^{-1/26})$, which is not very good at all. (See Figure 8.1.) Said another way, to get the same precision with p inputs that n data points gives us with one input takes $n^{(4+p)/5}$ data points. For $p = 100$, this is $n^{20.8}$, which tells us that matching the error of $n = 100$ one-dimensional observations requires $O(4 \times 10^{41})$ hundred-dimensional observations.

So completely unstructured nonparametric regressions won't work very well in high dimensions, at least not with plausible amounts of data. The trouble is that there are just *too many* possible high-dimensional functions, and seeing only a trillion points from the function doesn't pin down its shape very well at all.

[[ATTN: More mathematical explanation in appendix?]]

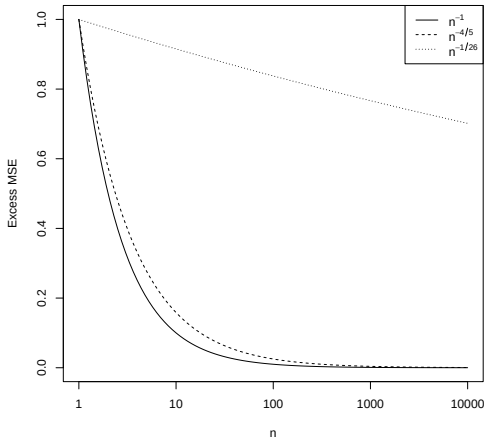
This is where additive models come in. Not every regression function is additive, so they have, even asymptotically, some approximation bias. But we can estimate each f_j by a simple one-dimensional smoothing, which converges at $O(n^{-4/5})$, almost as good as the parametric rate. So overall

$$MSE_{\text{additive}} - \sigma^2 = a_{\text{additive}} + O(n^{-4/5}) \tag{8.13}$$

Since linear models are a sub-class of additive models, $a_{\text{additive}} \leq a_{\text{lm}}$. From a purely predictive point of view, the only time to prefer linear models to additive models is when n is so small that $O(n^{-4/5}) - O(n^{-1})$ exceeds this difference in approximation biases; eventually the additive model will be more accurate.[6]

[5] Remember that in one dimension, the bias of a kernel smoother with bandwidth h is $O(h^2)$, and the variance is $O(1/nh)$, because only samples falling in an interval about h across contribute to the prediction at any one point, and when h is small, the number of such samples is proportional to nh. Adding bias squared to variance gives an error of $O(h^4) + O((nh)^{-1})$, solving for the best bandwidth gives $h_{\text{opt}} = O(n^{-1/5})$, and the total error is then $O(n^{-4/5})$. Suppose for the moment that in p dimensions we use the same bandwidth along each dimension. (We get the same end result with more work if we let each dimension have its own bandwidth.) The bias is still $O(h^2)$, because the Taylor expansion still goes through. But now only samples falling into a region of volume $O(h^p)$ around x contribute to the prediction at x, so the variance is $O((nh^p)^{-1})$. The best bandwidth is now $h_{\text{opt}} = O(n^{-1/(p+4)})$, yielding an error of $O(n^{-4/(p+4)})$ as promised.

[6] Unless the best additive approximation to μ is linear; then the linear model has no more bias and less variance.




```
curve(x^(-1),from=1,to=1e4,log="x",xlab="n",ylab="Excess MSE")
curve(x^(-4/5),add=TRUE,lty="dashed")
curve(x^(-1/26),add=TRUE,lty="dotted")
legend("topright",legend=c(expression(n^{-1}),
  expression(n^{-4/5}),expression(n^{-1/26})),
  lty=c("solid","dashed","dotted"))
```

Figure 8.1 Schematic of rates of convergence of MSEs for parametric models ($O(n^{-1})$), one-dimensional nonparametric regressions or additive models ($O(n^{-4/5})$), and a 100-dimensional nonparametric regression ($O(n^{-1/26})$). Note that the horizontal but not the vertical axis is on a logarithmic scale.

8.4 Example: California House Prices Revisited

As an example, we'll look at data on median house prices across Census tracts from the data-analysis assignment in §10. This has both California and Pennsylvania, but it's hard to visually see patterns with both states; I'll do California, and let you replicate this all on Pennsylvania, and even on the combined data.

Start with getting the data:

```
housing <- read.csv("http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/data/calif_penn_2011.csv")
housing <- na.omit(housing)
calif <- housing[housing$STATEFP == 6, ]
```

(How do I know that the `STATEFP` code of 6 corresponds to California?)

We'll fit a linear model for the log price, on the thought that it makes some sense for the factors which raise or lower house values to multiply together, rather than just adding.

```
calif.lm <- lm(log(Median_house_value) ~ Median_household_income + Mean_household_income +
    POPULATION + Total_units + Vacant_units + Owners + Median_rooms + Mean_household_size_owners +
    Mean_household_size_renters + LATITUDE + LONGITUDE, data = calif)
```

This is very fast — about a fifth of a second on my laptop.

Here are the summary statistics[7]:

```
print(summary(calif.lm), signif.stars = FALSE, digits = 3)
##
## Call:
## lm(formula = log(Median_house_value) ~ Median_household_income +
##     Mean_household_income + POPULATION + Total_units + Vacant_units +
##     Owners + Median_rooms + Mean_household_size_owners + Mean_household_size_renters +
##     LATITUDE + LONGITUDE, data = calif)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -3.855 -0.153  0.034  0.189  1.214
##
## Coefficients:
##                               Estimate Std. Error t value Pr(>|t|)
## (Intercept)                  -5.74e+00   5.28e-01  -10.86  < 2e-16
## Median_household_income       1.34e-06   4.63e-07    2.90   0.0038
## Mean_household_income         1.07e-05   3.88e-07   27.71  < 2e-16
## POPULATION                   -4.15e-05   5.03e-06   -8.27  < 2e-16
## Total_units                   8.37e-05   1.55e-05    5.41  6.4e-08
## Vacant_units                  8.37e-07   2.37e-05    0.04   0.9719
## Owners                       -3.98e-03   3.21e-04  -12.41  < 2e-16
## Median_rooms                 -1.62e-02   8.37e-03   -1.94   0.0525
## Mean_household_size_owners    5.60e-02   7.16e-03    7.83  5.8e-15
## Mean_household_size_renters -7.47e-02   6.38e-03  -11.71  < 2e-16
## LATITUDE                    -2.14e-01   5.66e-03  -37.76  < 2e-16
## LONGITUDE                   -2.15e-01   5.94e-03  -36.15  < 2e-16
##
## Residual standard error: 0.317 on 7469 degrees of freedom
```

[7] I have suppressed the usual stars on "significant" regression coefficients, because, as discussed in Chapter **??**, those aren't really the most important variables, and I have reined in R's tendency to use far too many decimal places.

```
predlims <- function(preds, sigma) {
    prediction.sd <- sqrt(preds$se.fit^2 + sigma^2)
    upper <- preds$fit + 2 * prediction.sd
    lower <- preds$fit - 2 * prediction.sd
    lims <- cbind(lower = lower, upper = upper)
    return(lims)
}
```

CODE EXAMPLE 19: *Calculating quick-and-dirty prediction limits from a prediction object (`preds`) containing fitted values and their standard errors, plus an estimate of the noise level. Because those are two (presumably uncorrelated) sources of noise, we combine the standard deviations by "adding in quadrature".*

```
## Multiple R-squared:  0.639,Adjusted R-squared:  0.638
## F-statistic: 1.2e+03 on 11 and 7469 DF,  p-value: <2e-16
```

Figure 8.2 plots the predicted prices, ± 2 standard errors, against the actual prices. The predictions are not all that *accurate* — the RMS residual is 0.317 on the log scale (i.e., 37% on the original scale), but they do have pretty reasonable coverage; about 96% of actual prices fall within the prediction limits[8]. On the other hand, the predictions *are* quite precise, with the median of the calculated standard errors being 0.011 on the log scale (i.e., 1.1% in dollars). This linear model *thinks* it knows what's going on.

Next, we'll fit an additive model, using the `gam` function from the `mgcv` package; this automatically decides how much smoothing is needed using a fast approximation to leave-one-out CV called **generalized cross-validation**, or **GCV** (§3.4.3).

```
system.time(calif.gam <- gam(log(Median_house_value) ~ s(Median_household_income) +
    s(Mean_household_income) + s(POPULATION) + s(Total_units) + s(Vacant_units) +
    s(Owners) + s(Median_rooms) + s(Mean_household_size_owners) + s(Mean_household_size_renters) +
    s(LATITUDE) + s(LONGITUDE), data = calif))
##    user  system elapsed
##   3.296   0.188   3.527
```

(That is, it took about five seconds total to run this.) The `s()` terms in the `gam` formula indicate which terms are to be smoothed — if we wanted particular parametric forms for some variables, we could do that as well. (Unfortunately we can't just write `MedianHouseValue ~ s(.)`, we have to list all the variables on

[8] Remember from your linear regression class that there are two kinds of confidence intervals we might want to use for prediction. One is a confidence interval for the *conditional mean* at a given value of x; the other is a confidence interval for the *realized values of* Y at a given x. Earlier examples have emphasized the former, but since we don't know the true conditional means here, we need to use the latter sort of intervals, prediction intervals proper, to evaluate coverage. The `predlims` function in Code Example 19 calculates a rough prediction interval by taking the standard error of the conditional mean, combining it with the estimated standard deviation, and multiplying by 2. Strictly speaking, we ought to worry about using a t-distribution rather than a Gaussian here, but with 7469 residual degrees of freedom, this isn't going to matter much. (Assuming Gaussian noise is likely to be more of a concern, but this is only meant to be a rough cut anyway.)

the right-hand side.[9]) The smoothing here is done by splines (hence `s()`), and there are lots of options for controlling the splines, or replacing them by other smoothers, if you know what you're doing.

Figure 8.3 compares the predicted to the actual responses. The RMS error has improved (0.27 on the log scale, or 130%, with 96% of observations falling with ± 2 standard errors of their fitted values), at only a fairly modest cost in the claimed precision (the median standard error of prediction is 0.02, or 2.1%). Figure 8.4 shows the partial response functions.

It makes little sense to have latitude and longitude make separate additive contributions here; presumably they interact. We can just smooth them together[10]:

```
calif.gam2 <- gam(log(Median_house_value) ~ s(Median_household_income) + s(Mean_household_income) +
    s(POPULATION) + s(Total_units) + s(Vacant_units) + s(Owners) + s(Median_rooms) +
    s(Mean_household_size_owners) + s(Mean_household_size_renters) + s(LONGITUDE,
    LATITUDE), data = calif)
```

This gives an RMS error of ± 0.25 (log-scale) and 96% coverage, with a median standard error of 0.021, so accuracy is improving (at least in sample), with little loss of precision.

Figures 8.6 and 8.7 show two different views of the joint smoothing of longitude and latitude. In the perspective plot, it's quite clear that price increases specifically towards the coast, and even more specifically towards the great coastal cities. In the contour plot, one sees more clearly an inward bulge of a negative, but not too very negative, contour line (between -122 and -120 longitude) which embraces Napa, Sacramento, and some related areas, which are comparatively more developed and more expensive than the rest of central California, and so more expensive than one would expect based on their distance from the coast and San Francisco.

If you worked through problem set 10, you will recall that one of the big things wrong with the linear model is that its errors (the residuals) are highly structured and very far from random. In essence, it totally missed the existence of cities, and the fact that houses cost more in cities (because land costs more there). It's a good idea, therefore, to make some maps, showing the actual values, and then, by way of contrast, the residuals of the models. Rather than do the plotting by hand over and over, let's write a function (Code Example 20).

Figures 8.8 and 8.9 show that allowing for the interaction of latitude and longitude (the smoothing term plotted in Figures 8.6–8.7) leads to a much more *random* and less systematic clumping of residuals. This is desirable in itself, even if it does little to improve the mean prediction error. Essentially, what that smoothing term is doing is picking out the existence of California's urban regions, and their distinction from the rural background. Examining the plots of the inter-

[9] Alternately, we could use Kevin Gilbert's `formulaTools` functions — see `https://gist.github.com/kgilbert-cmu`.

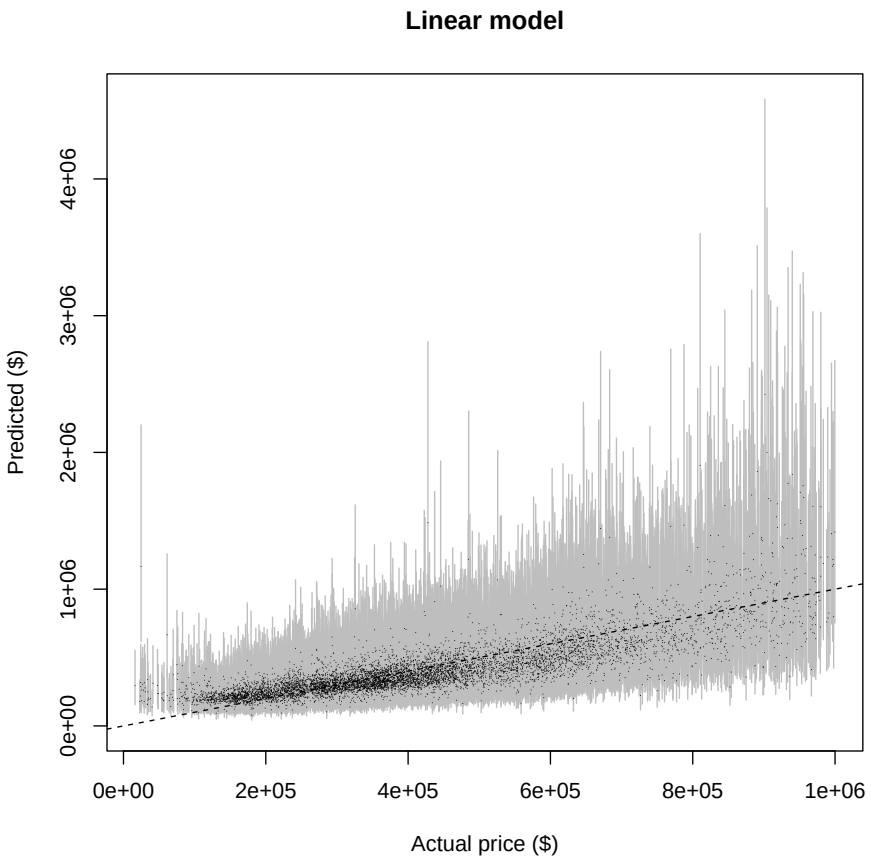
[10] If the two variables which interact have very different magnitudes, it's better to smooth them with a `te()` term than an `s()` term, but here they are comparable. See §8.5 for more, and `help(gam.models)`.

```
graymapper <- function(z, x = calif$LONGITUDE, y = calif$LATITUDE, n.levels = 10,
    breaks = NULL, break.by = "length", legend.loc = "topright", digits = 3, ...) {
    my.greys = grey(((n.levels - 1):0)/n.levels)
    if (!is.null(breaks)) {
        stopifnot(length(breaks) == (n.levels + 1))
    }
    else {
        if (identical(break.by, "length")) {
            breaks = seq(from = min(z), to = max(z), length.out = n.levels + 1)
        }
        else {
            breaks = quantile(z, probs = seq(0, 1, length.out = n.levels + 1))
        }
    }
    z = cut(z, breaks, include.lowest = TRUE)
    colors = my.greys[z]
    plot(x, y, col = colors, bg = colors, ...)
    if (!is.null(legend.loc)) {
        breaks.printable <- signif(breaks[1:n.levels], digits)
        legend(legend.loc, legend = breaks.printable, fill = my.greys)
    }
    invisible(breaks)
}
```

Code Example 20: *Map-making code. In its basic use, this takes vectors for x and y coordinates, and draws gray points whose color depends on a third vector for z, with darker points indicating higher values of z. Options allow for the control of the number of gray levels, setting the breaks between levels automatically, and using a legend. Returning the break-points makes it easier to use the same scale in multiple maps. See online for commented code.*

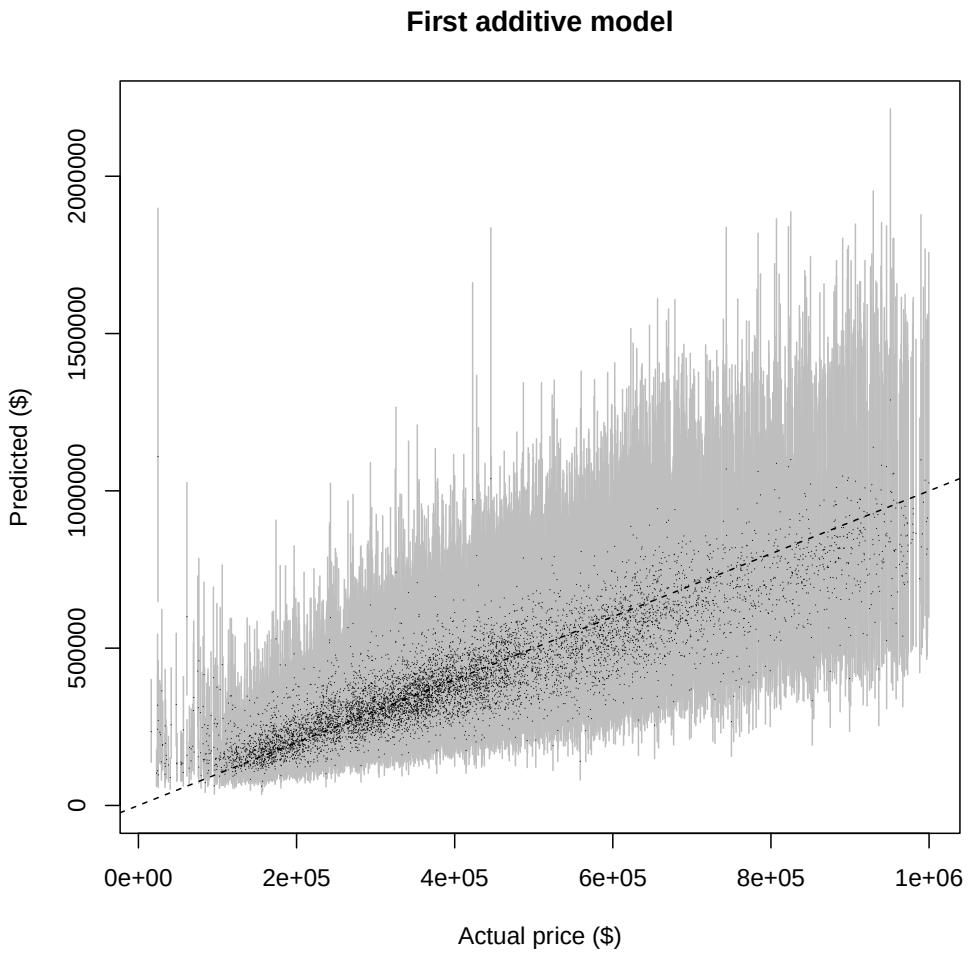
action term should suggest to you how inadequate it would be to just put in a `LONGITUDE`×`LATITUDE` term in a linear model.

Including an interaction between latitude and longitude in a spatial problem is pretty obvious. There are other potential interactions which might be important here — for instance, between the two measures of income, or between the total number of housing units available and the number of vacant units. We could, of course, just use a completely unrestricted nonparametric regression — going to the opposite extreme from the linear model. In addition to the possible curse-of-dimensionality issues, however, getting something like `npreg` to run with 7000 data points and 11 predictor variables requires a lot of patience. Other techniques, like nearest neighbor regression (§1.5.1) or regression trees (Ch. 13), may run faster, though cross-validation can be demanding even there.




```
plot(calif$Median_house_value, exp(preds.lm$fit), type = "n", xlab = "Actual price ($)",
    ylab = "Predicted ($)", main = "Linear model", ylim = c(0, exp(max(predlims.lm))))
segments(calif$Median_house_value, exp(predlims.lm[, "lower"]), calif$Median_house_value,
    exp(predlims.lm[, "upper"]), col = "grey")
abline(a = 0, b = 1, lty = "dashed")
points(calif$Median_house_value, exp(preds.lm$fit), pch = 16, cex = 0.1)
```

Figure 8.2 Actual median house values (horizontal axis) versus those predicted by the linear model (black dots), plus or minus two *predictive* standard errors (grey bars). The dashed line shows where actual and predicted prices are equal. Here `predict` gives both a fitted value for each point, and a standard error for that prediction. (Without a `newdata` argument, `predict` defaults to the data used to estimate `calif.lm`, which here is what we want.) Predictions are exponentiated so they're comparable to the original values (and because it's easier to grasp dollars than log-dollars).




```
plot(calif$Median_house_value, exp(preds.gam$fit), type = "n", xlab = "Actual price ($)",
    ylab = "Predicted ($)", main = "First additive model", ylim = c(0, exp(max(predlims.gam))))
segments(calif$Median_house_value, exp(predlims.gam[, "lower"]), calif$Median_house_value,
    exp(predlims.gam[, "upper"]), col = "grey")
abline(a = 0, b = 1, lty = "dashed")
points(calif$Median_house_value, exp(preds.gam$fit), pch = 16, cex = 0.1)
```

Figure 8.3 Actual versus predicted prices for the additive model, as in Figure 8.2. Note that the `sig2` attribute of a model returned by `gam()` is the estimate of the noise variance around the regression surface (σ^2).

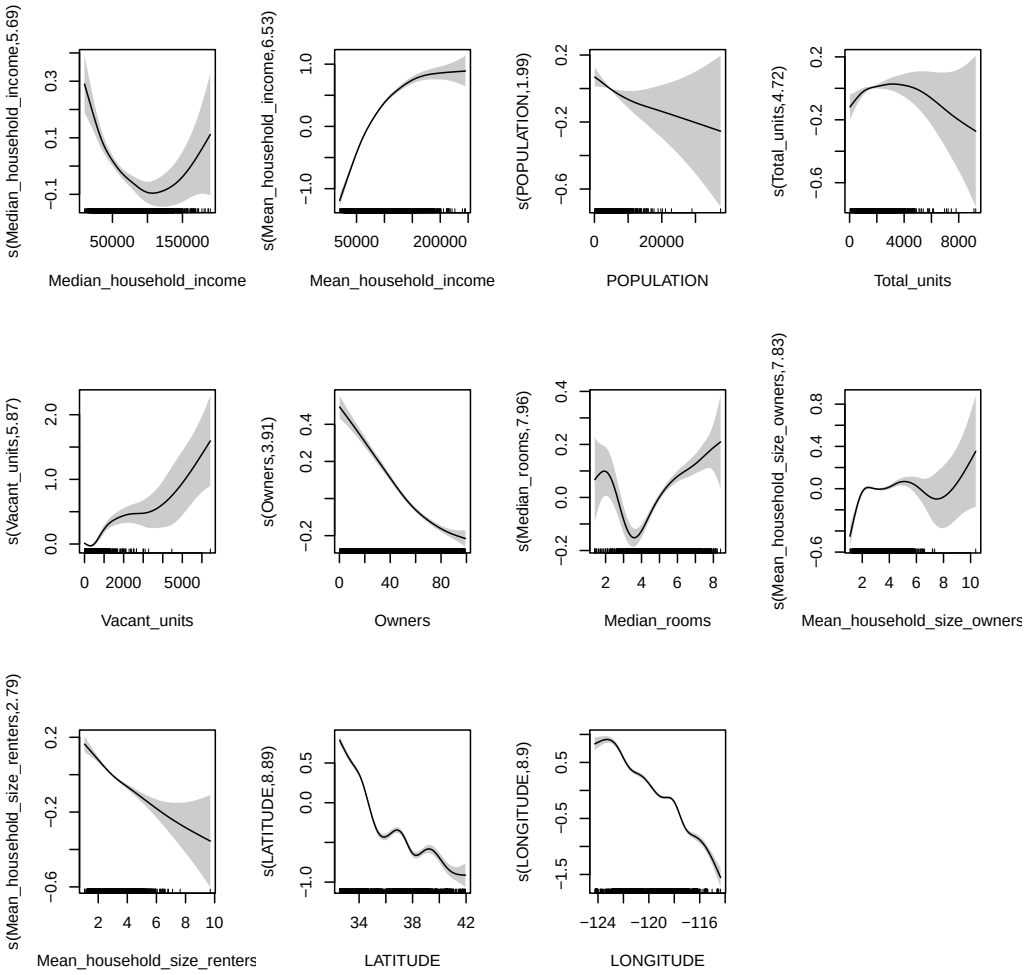
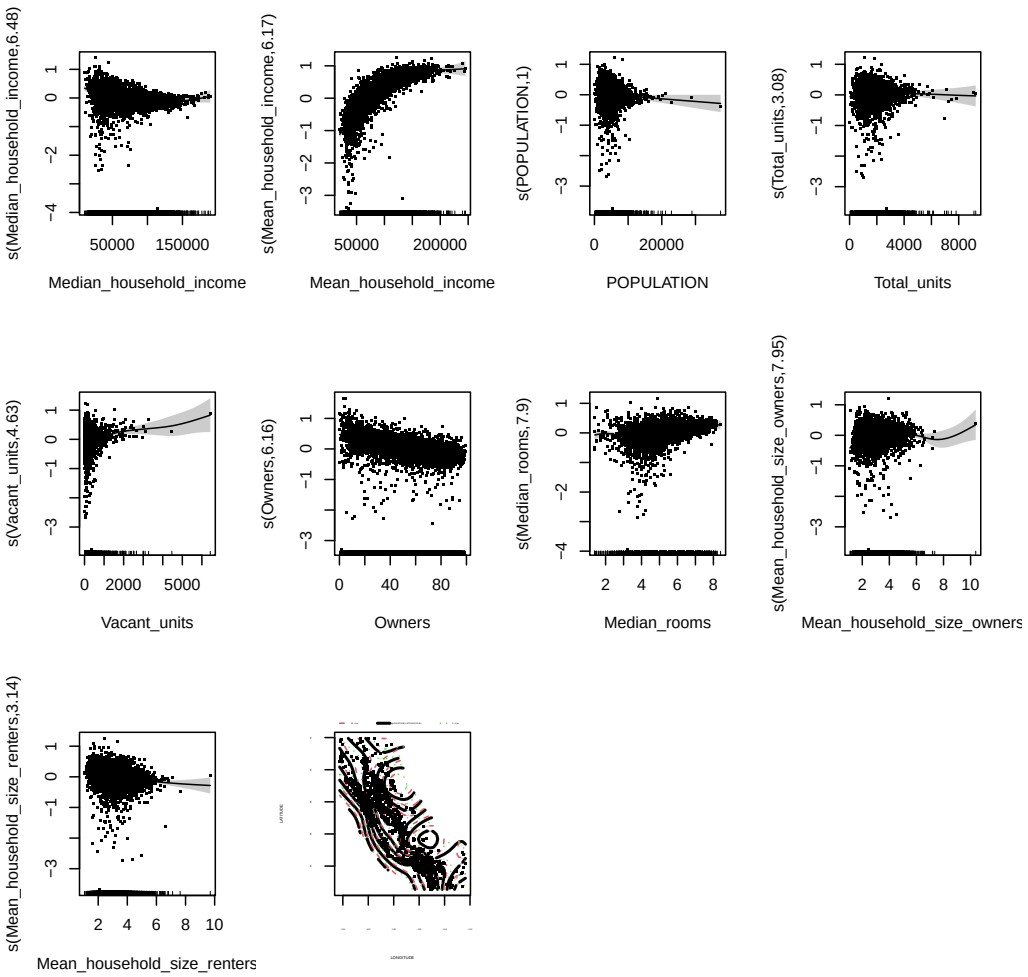
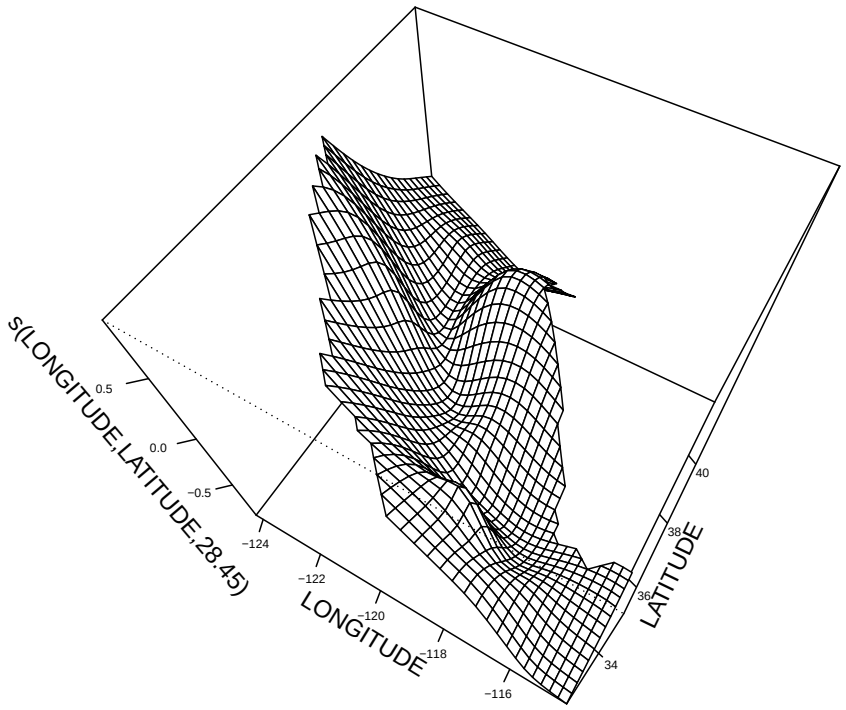


Figure 8.4 The estimated partial response functions for the additive model, with a shaded region showing ±2 standard errors. The tick marks along the horizontal axis show the observed values of the input variables (a **rug plot**); note that the error bars are wider where there are fewer observations. Setting `pages=0` (the default) would produce eight separate plots, with the user prompted to cycle through them. Setting `scale=0` gives each plot its own vertical scale; the default is to force them to share the same one. Finally, note that here the vertical scales are logarithmic.



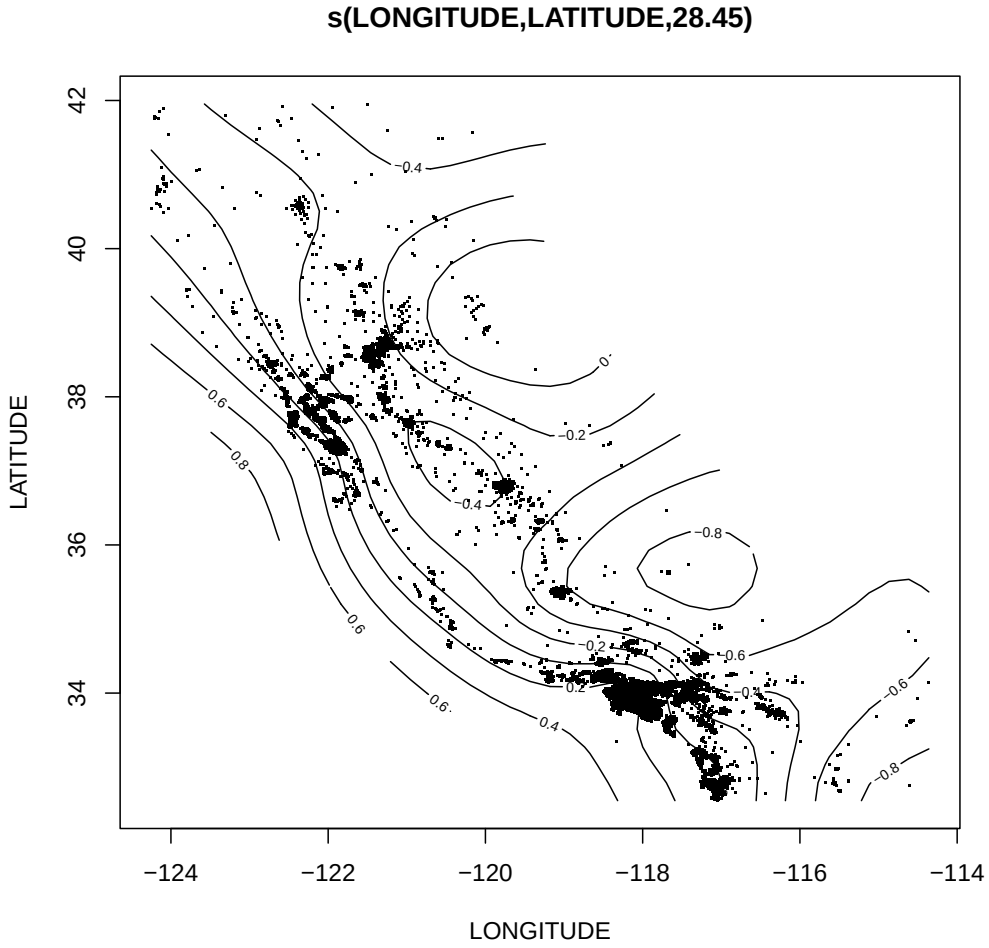

```
plot(calif.gam2, scale = 0, se = 2, shade = TRUE, resid = TRUE, pages = 1)
```

Figure 8.5 Partial response functions and partial residuals for `addfit2`, as in Figure 8.4. See subsequent figures for the joint smoothing of longitude and latitude, which here is an illegible mess. See `help(plot.gam)` for the plotting options used here.



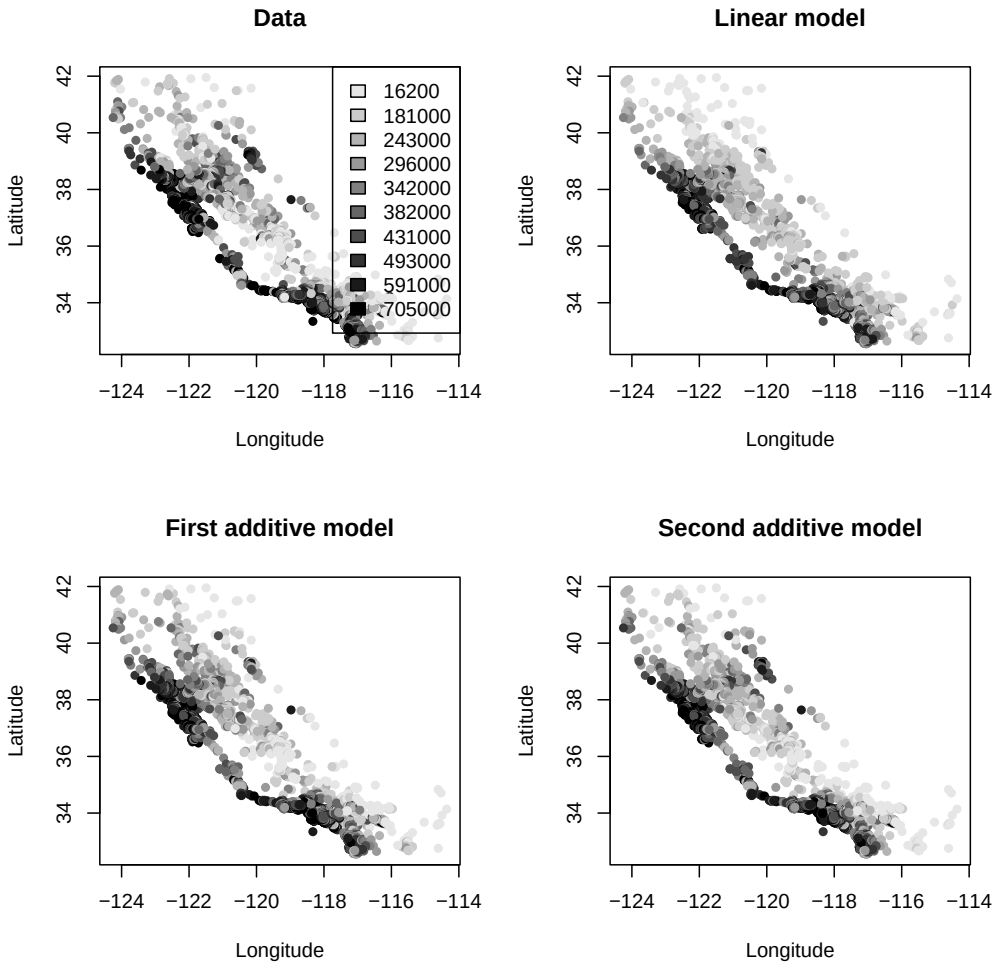

```
plot(calif.gam2, select = 10, phi = 60, pers = TRUE, ticktype = "detailed", cex.axis = 0.5)
```

Figure 8.6 The result of the joint smoothing of longitude and latitude.




```
plot(calif.gam2, select = 10, se = FALSE)
```

Figure 8.7 The result of the joint smoothing of longitude and latitude. Setting `se=TRUE`, the default, adds standard errors for the contour lines in multiple colors. Again, note that these are log units.




```
par(mfrow = c(2, 2))
calif.breaks <- graymapper(calif$Median_house_value, pch = 16, xlab = "Longitude",
    ylab = "Latitude", main = "Data", break.by = "quantiles")
graymapper(exp(preds.lm$fit), breaks = calif.breaks, pch = 16, xlab = "Longitude",
    ylab = "Latitude", legend.loc = NULL, main = "Linear model")
graymapper(exp(preds.gam$fit), breaks = calif.breaks, legend.loc = NULL, pch = 16,
    xlab = "Longitude", ylab = "Latitude", main = "First additive model")
graymapper(exp(preds.gam2$fit), breaks = calif.breaks, legend.loc = NULL, pch = 16,
    xlab = "Longitude", ylab = "Latitude", main = "Second additive model")
par(mfrow = c(1, 1))
```

Figure 8.8 Maps of real prices (top left), and those predicted by the linear model (top right), the purely additive model (bottom left), and the additive model with interaction between latitude and longitude (bottom right). Categories are deciles of the actual prices.

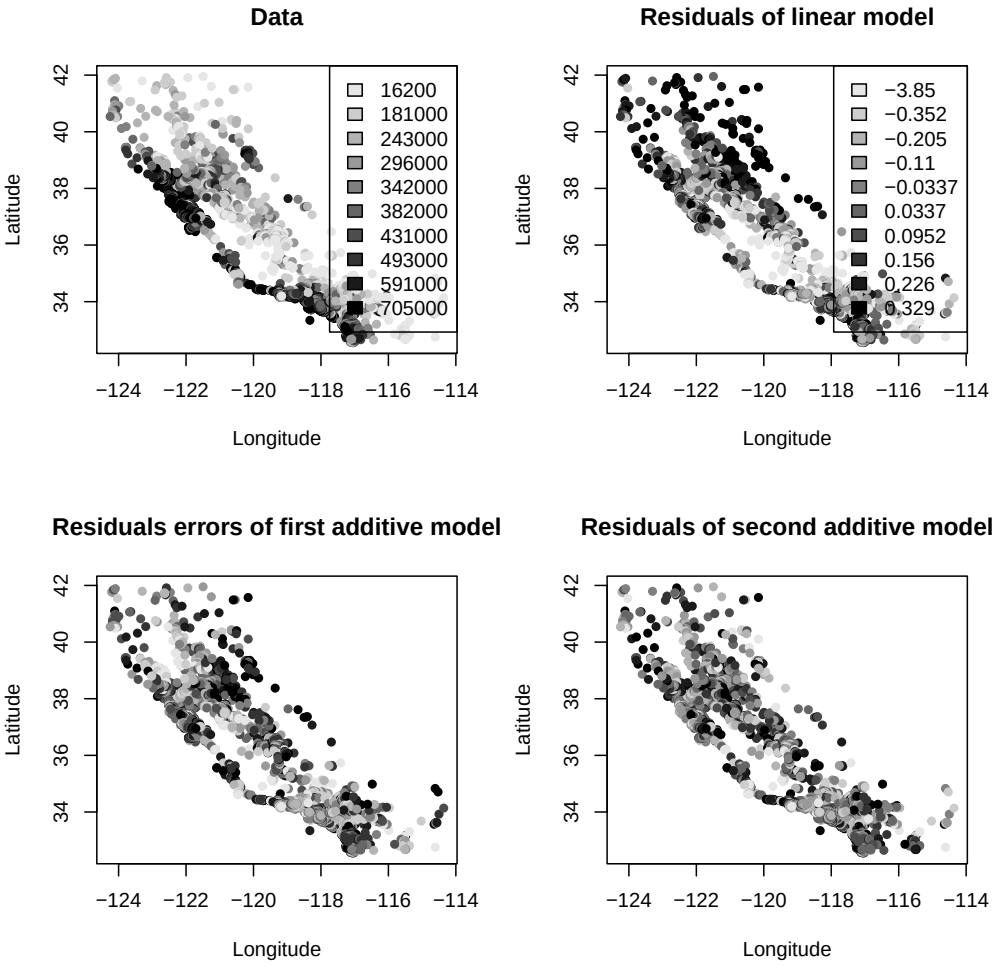


Figure 8.9 Actual housing values (top left), and the residuals of the three models. (The residuals are all plotted with the same color codes.) Notice that both the linear model and the additive model without spatial interaction systematically mis-price urban areas. The model with spatial interaction does much better at having randomly-scattered errors, though hardly perfect. — How would you make a map of the magnitude of regression errors?

8.5 Interaction Terms and Expansions

One way to think about additive models, and about (possibly) including interaction terms, is to imagine doing a sort of Taylor series or power series expansion of the true regression function. The zero-th order expansion would be a constant:

$$\mu(x) \approx \alpha \tag{8.14}$$

The best constant to use here would just be $\mathbb{E}[Y]$. ("Best" here is in the mean-square sense, as usual.) A purely additive model would correspond to a first-order expansion:

$$\mu(x) \approx \alpha + \sum_{j=1}^{p} f_j(x_j) \tag{8.15}$$

Two-way interactions come in when we go to a second-order expansion:

$$\mu(x) \approx \alpha + \sum_{j=1}^{p} f_j(x_j) + \sum_{j=1}^{p} \sum_{k=j+1}^{p} f_{jk}(x_j, x_k) \tag{8.16}$$

(Why do I limit k to run from $j+1$ to p?, rather than from 1 to p?) We will, of course, insist that $\mathbb{E}[f_{jk}(X_j, X_k)] = 0$ for all j, k. If we want to estimate these terms in R, using `mgcv`, we use the syntax `s(xj, xk)` or `te(xj, xk)`. The former fits a thin-plate spline over the (x_j, x_k) plane, and is appropriate when those variables are measured on similar scales, so that curvatures along each direction are comparable. The latter uses a tensor product of smoothing splines along each coordinate, and is more appropriate when the measurement scales are very different[11].

There is an important ambiguity here: for any j, with additive partial-response function f_j, I could take any of its interactions, set $f'_{jk}(x_j, x_k) = f_{jk}(x_j, x_k) + f_j(x_j)$ and $f'_j(x_j) = 0$, and get exactly the same predictions under all circumstances. This is the parallel to being able to add and subtract constants from the first-order functions, provided we made corresponding changes to the intercept term. We therefore need to similarly fix the two-way interaction functions.

A natural way to do this is to insist that the second-order f_{jk} function should be uncorrelated with ("orthogonal to") the first-order functions f_j and f_k; this is the analog to insisting that the first-order functions all have expectation zero. The f_{jk}s then represent purely interactive contributions to the response, which could not be captured by additive terms. If this is what we want to do, the best syntax to use in `mgcv` is `ti`, which specifically separates the first- and higher-order terms, e.g., `ti(xj) + ti(xk) + ti(xj, xk)` will estimate three functions, for the additive contributions and their interaction.

An alternative is to just *pick* a particular f_{jk}, and absorb f_j into it. The model

[11] For the distinction between thin-plate and tensor-product splines, see §7.4. If we want to interact a continuous variable x_j with a categorical x_k, `mgcv`'s syntax is `s(xj, by=xk)` or `te(xj, by=xk)`.

then looks like

$$\mu(x) \approx \alpha + \sum_{j=1}^{p} \sum_{k=j+1}^{p} f_{jk}(x_j, x_k) \tag{8.17}$$

We can also mix these two approaches, if we specifically do not want additive or interactive terms for certain predictor variables. This is what I did above, where I estimated a single second-order smoothing term for both latitude and longitude, with no additive components for either.

Of course, there is nothing special about two-way interactions. If you're curious about what a three-way term would be like, and you're lucky enough to have data which amenable to fitting it, you could certainly try

$$\mu \approx \alpha + \sum_{j=1}^{p} f_j(x_j) + \sum_{j=1}^{p} \sum_{k=j+1}^{p} f_{jk}(x_j, x_k) + \sum_{j,k,l} f_{jkl}(x_j, x_k, x_l) \tag{8.18}$$

(How should the indices for the last term go?) More ambitious combinations are certainly possible, though they tend to become a confused mass of algebra and indices.

Geometric interpretation

It's often convenient to think of the regression function as living in a big (infinite-dimensional) vector space of functions. Within this space, the *constant* functions form a linear sub-space[12], and we can ask for the projection of the true regression function on to that sub-space; this would be the best approximation[13] to μ as a constant. This is, of course, the expectation value. The *additive* functions of all p variables also form a linear sub-space[14], so the right-hand side of Eq. 8.15 is just the projection of μ on to that space, and so forth and so on. When we insist on having the higher-order interaction functions be uncorrelated with the additive functions, we're taking the projection of μ on to the space of all functions *orthogonal* to the additive functions.

Selecting interactions

There are two issues with interaction terms. First, the curse of dimensionality returns: an order-q interaction term will converge at the rate $O(n^{-4/(4+q)})$, so they can dominate the over-all uncertainty. Second, there are lots of possible interactions ($\binom{p}{q}$, in fact), which can make it very demanding in time and data to fit them all, and hard to interpret. Just as with linear models, therefore, it can make a lot of sense to selectively include interactions based on subject-matter knowledge, or examination of residuals of additive models.

[12] Because if f and g are two constant functions, $af + bg$ is also a constant, for any real numbers a and b.

[13] Remember that projecting a vector on to a linear sub-space finds the point in the sub-space closest to the original vector. This is equivalent to minimizing the (squared) bias.

[14] By parallel reasoning to the previous footnote.

Varying-coefficient models

In some contexts, people like to use models of the form

$$\mu(x) = \alpha + \sum_{j=1}^{p} x_j f_j(x_{-j}) \tag{8.19}$$

where f_j is a function of the non-j predictor variables, or some subset of them. These **varying-coefficient** functions are obviously a subset of the usual class of additive models, but there are occasions where they have some scientific justification[15]. These are conveniently estimated in `mgcv` through the `by` option, e.g., `s(xk, by=xj)` will estimate a term of the form $x_j f(x_k)$.[16]

8.6 Closing Modeling Advice

With modern computing power, there are very few situations in which it is actually better to do linear regression than to fit an additive model. In fact, there seem to be only two good reasons to prefer linear models.

1. Our data analysis is guided by a credible scientific theory which asserts linear relationships *among the variables we measure* (not others, for which our observables serve as imperfect proxies).
2. Our data set is so massive that either the extra processing time, or the extra computer memory, needed to fit and store an additive rather than a linear model is prohibitive.

Even when the first reason applies, and we have good reasons to believe a linear theory, the truly scientific thing to do would be to *check* linearity, by fitting a flexible non-linear model and seeing if it looks close to linear. (We will see formal tests based on this idea in Chapter 9.) Even when the second reason applies, we would like to know how much bias we're introducing by using linear predictors, which we could do by randomly selecting a subset of the data which is small enough for us to manage, and fitting an additive model.

In the vast majority of cases when users of statistical software fit linear models, neither of these justifications applies: theory doesn't tell us to expect linearity, and our machines don't compel us to use it. Linear regression is then employed for no better reason than that users know how to type `lm` but not `gam`. *You* now know better, and can spread the word.

8.7 Further Reading

Simon Wood, who wrote the `mgcv` package, has a nice book about additive models and their generalizations, Wood (2006); at this level it's your best source for further information. Buja *et al.* (1989) dives further into some theoretical issues.

[15] They can also serve as a "transitional object" when giving up the use of purely linear models.

[16] As we saw above, `by` does something slightly different when given a categorical variable. How are these two uses related?

The expansions of §8.5 are sometimes called "functional analysis of variance" or "functional ANOVA". Making those ideas precise requires exploring some of the geometry of infinite-dimensional spaces of functions ("Hilbert space"). See Wahba (1990) for a treatment of the statistical topic, and Halmos (1957) for a classic introduction to Hilbert spaces.

Historical notes

Ezekiel (1924) seems to be the first publication advocating the use of additive models as a general method, which he called "curvilinear multiple correlation". His paper was complete with worked examples on simulated data (with known answers) and real data (from economics)[17]. He was explicit that any reasonable smoothing or regression technique could be used to find what we'd call the partial response functions. He also gave a successive-approximation algorithm for estimate the over-all model: start with an initial guess about all the partial responses; plot all the partial residuals; refine the partial responses simultaneously; repeat. This differs from back-fitting in that the partial response functions are updating in parallel within each cycle, not one after the other. This is a subtle difference, and Ezekiel's method will often work, but can run into trouble with correlated predictor variables, when back-fitting will not.

The Gauss-Seidel or backfitting algorithm was invented by Gauss in the early 1800s during his work on least squares estimation in linear models; he mentioned it in letters to students, described it as something one could do "while half asleep", but never published it. Seidel gave the first published version in 1874. (For this history, see Benzi 2009.) I am not sure when the connection was made between additive statistical models and back-fitting.

Exercises

8.1 Repeat the analyses of California housing prices with Pennsylvania housing prices. Which partial response functions might one reasonably hope would stay the same? Do they? (How can you tell?)

8.2 *Additive?* For general p, let $\|\vec{x}\|$ be the (ordinary, Euclidean) length of the vector $\vec{x}$. Is this an additive function of the (ordinary, Cartesian) coordinates? Is $\|\vec{x}\|^2$ an additive function? $\|\vec{x} - \vec{x}_0\|$ for a fixed $\vec{x}_0$? $\|\vec{x} - \vec{x}_0\|^2$?

8.3 *Additivity vs. parallelism*

1. Take any additive function f of p arguments $x_1, x_2, \ldots x_p$. Fix a coordinate index i and a real number c. Prove that $f(x_1, x_2, \ldots x_i, \ldots x_p) - f(x_1, x_2, \ldots x_i + c, \ldots x_p)$ depends only on x_i and c, and not on the other coordinates.
2. Suppose $p = 2$, and continue to assume f is additive. Consider the curve formed by plotting $f(x_1, x_2)$ against x_1 for a fixed value of x_2, and the curved formed by plotting

[17] "Each of these curves illustrates and substantiates conclusions reached by theoretical economic analysis. Equally important, they provide definite quantitative statements of the relationships. The method of ... curvilinear multiple correlation enable[s] us to use the favorite tool of the economist, *caeteris paribus*, in the analysis of actual happenings equally as well as in the intricacies of theoretical reasoning" (p. 453). (See also Exercise 8.4.)

$f(x_1, x_2)$ against x_1 with x_2 fixed at a different value, say x_2'. Prove that the curves are parallel, i.e., that the vertical distance between them is constant.

3. For general p and additive f, consider the surfaces formed by the f by varying all but one of the coordinates. Prove that these surfaces are always parallel to each other.
4. Is the converse true? That is, do parallel regression surfaces imply an additive model?

8.4 *Additivity vs. partial derivatives*

1. Suppose that the true regression function μ is additive, with partial response functions f_j. Show that $\frac{\partial \mu}{\partial x_j} = f_j(x_j)$, so that each partial derivative is a function of that coordinate alone.
2. *(Much harder)* Suppose that, for each coordinate x_j, there is some function f_j of x_j alone such that $\frac{\partial \mu}{\partial x_j} = f_j(x_j)$. Is μ necessarily additive?

8.5 Suppose that an additive model holds, so that $Y = \alpha + \sum_{j=1}^{p} f_j(X_j) + \epsilon$, with $\alpha = \mathbb{E}[Y]$, $\mathbb{E}\left[f_j(X_j)\right] = 0$ for each j, and $\mathbb{E}[\epsilon | X = x] = 0$ for all x.

1. For each j, let $\mu_j(x_j) = \mathbb{E}\left[Y | X_j = x_j\right]$. Show that

$$\mu_j(x_j) = \alpha + f_j(x_j) + \sum_{k \neq j} \mathbb{E}\left[f_k(X_k) | X_j = x_j\right]$$

2. Show that if X_k is statistically independent of X_j, for all $k \neq j$, then $\mu_j(x_j) - \alpha = f_j(x_j)$.
3. Does the conclusion of Exercise 22 still hold if one or more of the X_ks is statistically dependent on X_j? Explain why this should be the case, or give a counter-example to show that it's not true. *Hint:* All linear models are additive models, so if it is true for all additive models, it's true for all linear models. *Is* it true for all linear models?

9

Testing Parametric Regression Specifications with Nonparametric Regression

9.1 Testing Functional Forms

An important, if under-appreciated, use of nonparametric regression is checking whether parametric regressions are well-specified. The typical parametric regression model is something like

$$Y = f(X; \theta) + \epsilon \tag{9.1}$$

where f is some function which is completely specified except for the finite vector of parameters θ, and ϵ, as usual, is uncorrelated noise. Often, of course, people use a function f that is linear in the variables in X, or perhaps includes some interactions between them.

How can we tell if the specification is right? If, for example, it's a linear model, how can we check whether there might not be some nonlinearity? A common approach is to modify the specification to allow for *specific* departures from the baseline model — say, adding a quadratic term — and seeing whether the coefficients that go with those terms are significantly non-zero, or whether the improvement in fit is significant.[1] For example, one might compare the model

$$Y = \theta_1 x_1 + \theta_2 x_2 + \epsilon \tag{9.2}$$

to the model

$$Y = \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \epsilon \tag{9.3}$$

by checking whether the estimated θ_3 is significantly different from 0, or whether the residuals from the second model are significantly smaller than the residuals from the first.

This can work, *if* you have chosen the right nonlinearity to test. It has the power to detect certain mis-specifications, if they exist, but not others. (What if the departure from linearity is not quadratic but cubic?) If you have good reasons to think that when the model is wrong, it can only be wrong in certain ways, fine; if not, though, why only check for those errors?

Nonparametric regression effectively lets you check for *all* kinds of systematic errors, rather than singling out a particular one. There are three basic approaches, which I give in order of increasing sophistication.

[1] In my experience, this approach is second in popularity only to ignoring the issue.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

- If the parametric model is right, it should predict as well as, or even better than, the non-parametric one, and we can check whether $MSE_p(\widehat{\theta}) - MSE_{np}(\widehat{\mu})$ is sufficiently small.
- If the parametric model is right, the non-parametric estimated regression curve should be very close to the parametric one. So we can check whether $f(x;\widehat{\theta}) - \widehat{\mu}(x)$ is approximately zero everywhere.
- If the parametric model is right, then its *residuals* should be patternless and independent of input features, because

$$\mathbb{E}\left[Y - f(x;\theta)|X\right] = \mathbb{E}\left[f(x;\theta) + \epsilon - f(x;\theta)|X\right] = \mathbb{E}\left[\epsilon|X\right] = 0 \tag{9.4}$$

  So we can apply non-parametric smoothing to the parametric residuals, $y - f(x;\widehat{\theta})$, and see if their expectation is approximately zero everywhere.

We'll stick with the first procedure, because it's simpler for us to implement computationally. However, it turns out to be easier to develop theory for the other two, and especially for the third — see Li and Racine (2007, ch. 12), or Hart (1997).

Here is the basic procedure.

1. Get data $(x_1, y_1), (x_2, y_2), \ldots (x_n, y_n)$.
2. Fit the parametric model, getting an estimate $\widehat{\theta}$, and in-sample mean-squared error $MSE_p(\widehat{\theta})$.
3. Fit your favorite nonparametric regression (using cross-validation to pick control settings as necessary), getting curve $\widehat{\mu}$ and in-sample mean-squared error $MSE_{np}(\widehat{\mu})$.
4. Calculate $\widehat{d} = MSE_p(\widehat{\theta}) - MSE_{np}(\widehat{\mu})$.
5. Simulate from the parametric model $\widehat{\theta}$ to get faked data $(x_1^*, y_1^*), \ldots (x_n^*, y_n^*)$.
   1. Fit the parametric model to the simulated data, getting estimate $\widehat{\theta}^*$ and $MSE_p(\widehat{\theta}^*)$.
   2. Fit the nonparametric model to the simulated data, getting estimate $\widehat{\mu}^*$ and $MSE_{np}(\widehat{\mu}^*)$.
   3. Calculate $D^* = MSE_p(\widehat{\theta}^*) - MSE_{np}(\widehat{\mu}^*)$.
6. Repeat step 5 b times to get an estimate of the distribution of D under the null hypothesis.
7. The approximate p-value is $\frac{1+\#\left\{D^* > \widehat{d}\right\}}{1+b}$.

Let's step through the logic. In general, the error of the non-parametric model will be converging to the smallest level compatible with the intrinsic noise of the process. What about the parametric model?

Suppose on the one hand that the parametric model *is* correctly specified. Then its error will also be converging to the minimum — by assumption, it's got the functional form right so bias will go to zero, and as $\widehat{\theta} \to \theta_0$, the variance will also

go to zero. In fact, with enough data the correctly-specified parametric model will actually *generalize* better than the non-parametric model[2].

Suppose on the other hand that the parametric model is mis-specified. Then its predictions are systematically wrong, even with unlimited amounts of data — there's some bias which never goes away, no matter how big the sample. Since the non-parametric smoother does eventually come arbitrarily close to the true regression function, the smoother will end up predicting better than the parametric model.

Smaller errors for the smoother, then, suggest that the parametric model is wrong. But since the smoother has higher capacity, it could easily get smaller errors on a particular sample by chance and/or over-fitting, so only *big* differences in error count as evidence. Simulating from the parametric model gives us surrogate data which looks just like reality ought to, *if* the model is true. We then see how much better we could expect the non-parametric smoother to fit *under the parametric model*. If the non-parametric smoother fits the actual data much better than this, we can reject the parametric model with high confidence: it's really unlikely that we'd see that big an improvement from using the nonparametric model just by luck.[3]

As usual, we simulate from the parametric model simply because we have no hope of working out the distribution of the differences in MSEs from first principles. This is an example of our general strategy of bootstrapping.

[2] Remember that the smoother must, so to speak, use up some of the information in the data to figure out the shape of the regression function. The parametric model, on the other hand, takes that basic shape as given, and uses all the data's information to tune its parameters.

[3] As usual with p-values, this is not symmetric. A high p-value might mean that the true regression function is very close to $\mu(x;\theta)$, or it might mean that we don't have enough data to draw conclusions (or that we were unlucky).

9.1.1 Examples of Testing a Parametric Model

Let's see this in action. First, let's detect a reasonably subtle nonlinearity. Take the non-linear function $g(x) = \log(1+x)$, and say that $Y = g(x) + \epsilon$, with ϵ being IID Gaussian noise with mean 0 and standard deviation 0.15. (This is one of the examples from §4.2.) Figure 9.1 shows the regression function and the data. The nonlinearity is clear with the curve to "guide the eye", but fairly subtle.

A simple linear regression looks pretty good:

```
glinfit = lm(y ~ x, data = gframe)
print(summary(glinfit), signif.stars = FALSE, digits = 2)
##
## Call:
## lm(formula = y ~ x, data = gframe)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -0.468 -0.109 -0.002  0.114  0.485
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    0.218      0.021      11   <2e-16
## x              0.420      0.012      36   <2e-16
##
## Residual standard error: 0.18 on 298 degrees of freedom
## Multiple R-squared:  0.81,Adjusted R-squared:  0.81
## F-statistic: 1.3e+03 on 1 and 298 DF,  p-value: <2e-16
```

R^2 is ridiculously high — the regression line preserves 81 percent of the variance in the data. The p-value reported by R is also very, very low, but remember all this really means is "you'd have to be crazy to think a flat line fit better than straight line with a slope" (Figure 9.2).

The in-sample MSE of the linear fit is[4]

```
signif(mean(residuals(glinfit)^2), 3)
## [1] 0.0311
```

The nonparametric regression has a somewhat smaller MSE[5]

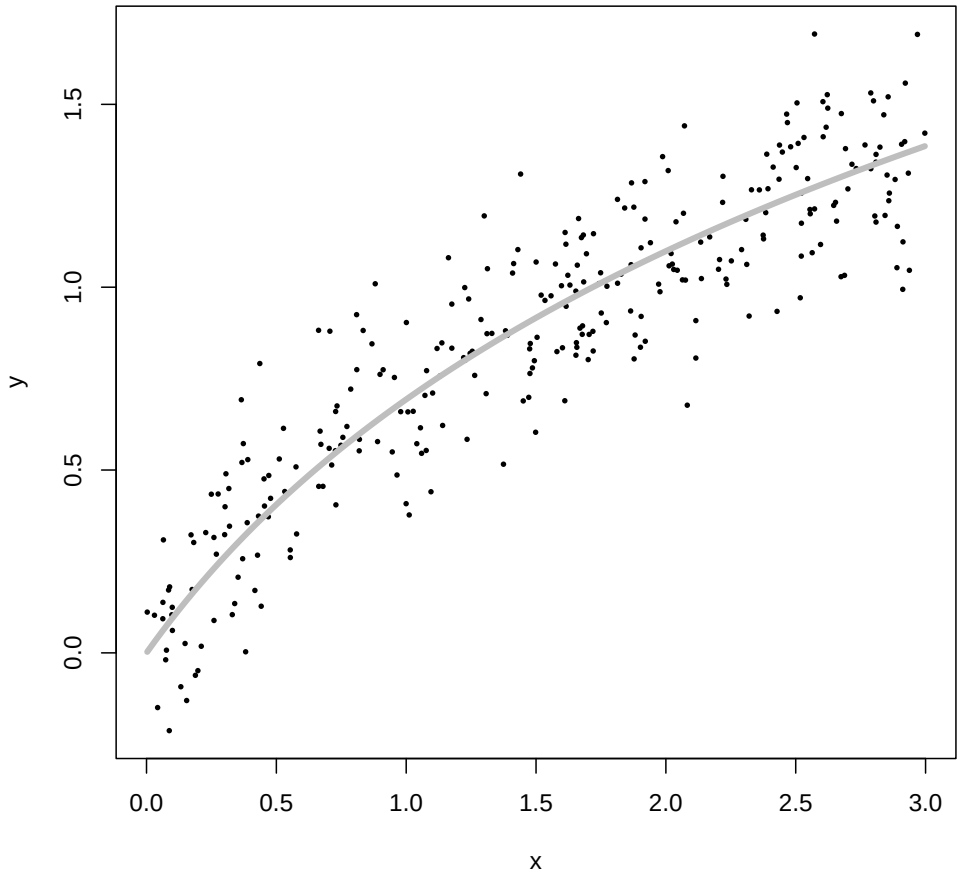
```
library(np)
gnpr <- npreg(y ~ x, data = gframe)
signif(gnpr$MSE, 3)
## [1] 0.024
```

So $\widehat{d}$ is

```
signif((d.hat = mean(glinfit$residual^2) - gnpr$MSE), 3)
## [1] 0.00715
```

[4] If we ask R for the MSE, by squaring `summary(glinfit)$sigma`, we get 0.031346. This differs from the mean of the squared residuals by a factor of factor of $n/(n-2) = 300/298 = 1.0067$, because R is trying to estimate the out-of-sample error by scaling up the in-sample error, the same way the estimated population variance scales up the sample variance. We want to compare in-sample fits.

[5] `npreg` does not apply the kind of correction mentioned in the previous footnote.




```
x <- runif(300, 0, 3)
yg <- log(x + 1) + rnorm(length(x), 0, 0.15)
gframe <- data.frame(x = x, y = yg)
plot(x, yg, xlab = "x", ylab = "y", pch = 16, cex = 0.5)
curve(log(1 + x), col = "grey", add = TRUE, lwd = 4)
```

Figure 9.1 True regression curve (grey) and data points (circles). The curve $g(x) = \log(1 + x)$.

Now we need to simulate from the fitted parametric model, using its estimated coefficients and noise level. We have seen several times now how to do this. The function `sim.lm` in Example 21 does this, along the same lines as the examples in Chapter 6; it assumes homoskedastic Gaussian noise. Again, as before, we need

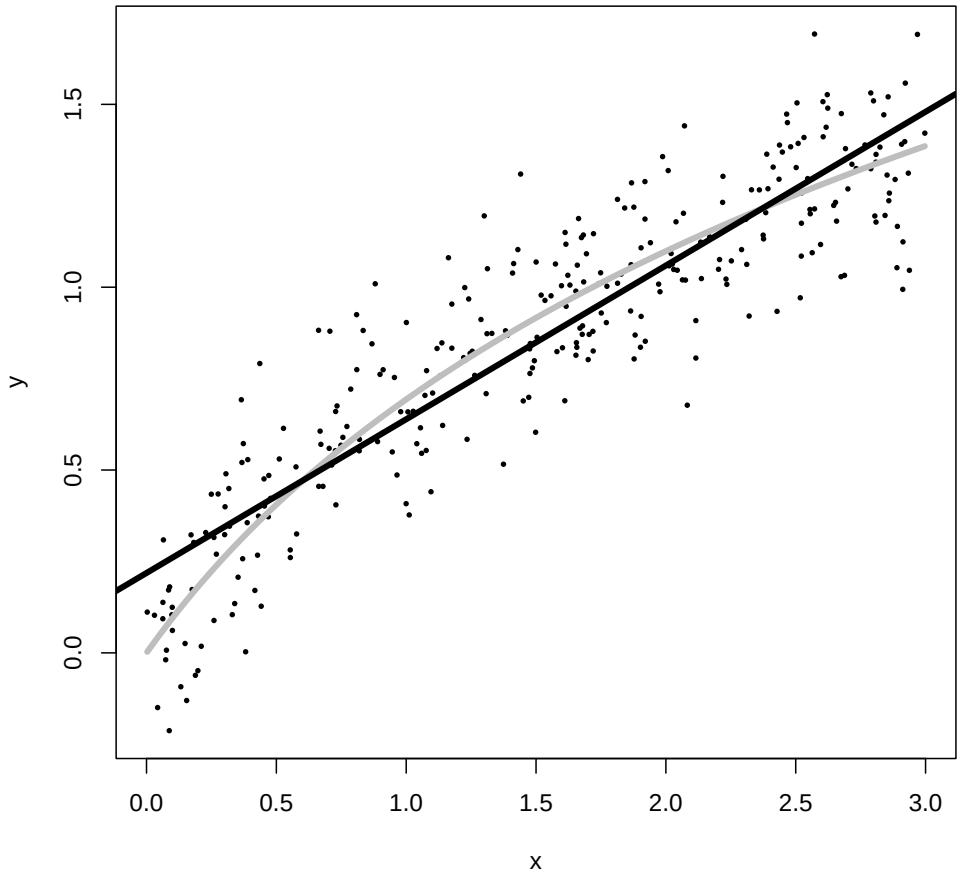


Figure 9.2 As in previous figure, but adding the least-squares regression line (black). Line widths exaggerated for clarity.

a function which will calculate the difference in MSEs between a linear model and a kernel smoother fit to the same data set — which will do automatically what we did by hand above. This is `calc.D` in Example 22. Note that the kernel bandwidth has to be re-tuned to each new data set.

If we call `calc.D` on the output of `sim.lm`, we get one value of the test statistic under the null distribution:

```
calc.D(sim.lm(glinfit, x))
## [1] 0.002131516
```

Now we just repeat this a lot to get a good approximation to the sampling distribution of D under the null hypothesis:

```
sim.lm <- function(linfit, test.x) {
    n <- length(test.x)
    sim.frame <- data.frame(x = test.x)
    sigma <- summary(linfit)$sigma * (n - 2)/n
    y.sim <- predict(linfit, newdata = sim.frame)
    y.sim <- y.sim + rnorm(n, 0, sigma)
    sim.frame <- data.frame(sim.frame, y = y.sim)
    return(sim.frame)
}
```

Code Example 21: *Simulate a new data set from a linear model, assuming homoskedastic Gaussian noise. It also assumes that there is one input variable,* `x`*, and that the response variable is called* `y`*. Could you modify it to work with multiple regression?*

```
calc.D <- function(data) {
    MSE.p <- mean((lm(y ~ x, data = data)$residuals)^2)
    MSE.np.bw <- npregbw(y ~ x, data = data)
    MSE.np <- npreg(MSE.np.bw)$MSE
    return(MSE.p - MSE.np)
}
```

Code Example 22: *Calculate the difference-in-MSEs test statistic.*

```
null.samples.D <- replicate(200, calc.D(sim.lm(glinfit, x)))
```

This takes some time, because each replication involves not just generating a new simulation sample, but also cross-validation to pick a bandwidth. This adds up to about a second per replicate on my laptop, and so a couple of minutes for 200 replicates.

(While the computer is thinking, look at the command a little more closely. It leaves the `x` values alone, and only uses simulation to generate new y values. This is appropriate here because our model doesn't really *say* where the `x` values came from; it's just about the conditional distribution of Y *given* X. If the model we were testing specified a distribution for x, we should generate x each time we invoke `calc.D`. If the specification is vague, like "x is IID" but with no particular distribution, then resample X.)

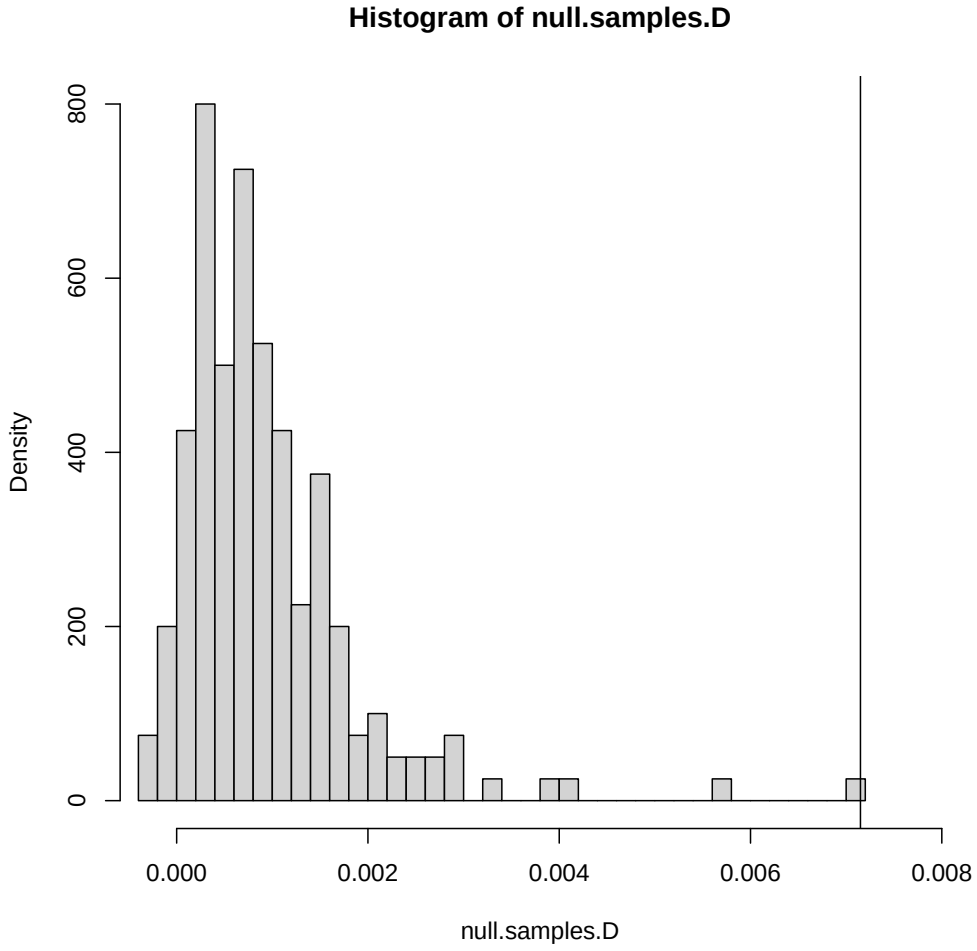
When it's done, we can plot the distribution and see that the observed value $\widehat{d}$ is pretty far out along the right tail (Figure 9.3). This tells us that it's very unlikely that `npreg` would improve so much on the linear model if the latter were true. In fact, exactly 0 of the simulated values of the test statistic were that big:

```
sum(null.samples.D > d.hat)
## [1] 0
```

Thus our estimated p-value is ≤ 0.00498. We can reject the linear model pretty confidently.[6]

As a second example, let's suppose that the linear model *is* right — then the

[6] If we wanted a more precise estimate of the p-value, we'd need to use more bootstrap samples.




```
hist(null.samples.D, n = 31, xlim = c(min(null.samples.D), 1.1 * d.hat), probability = TRUE)
abline(v = d.hat)
```

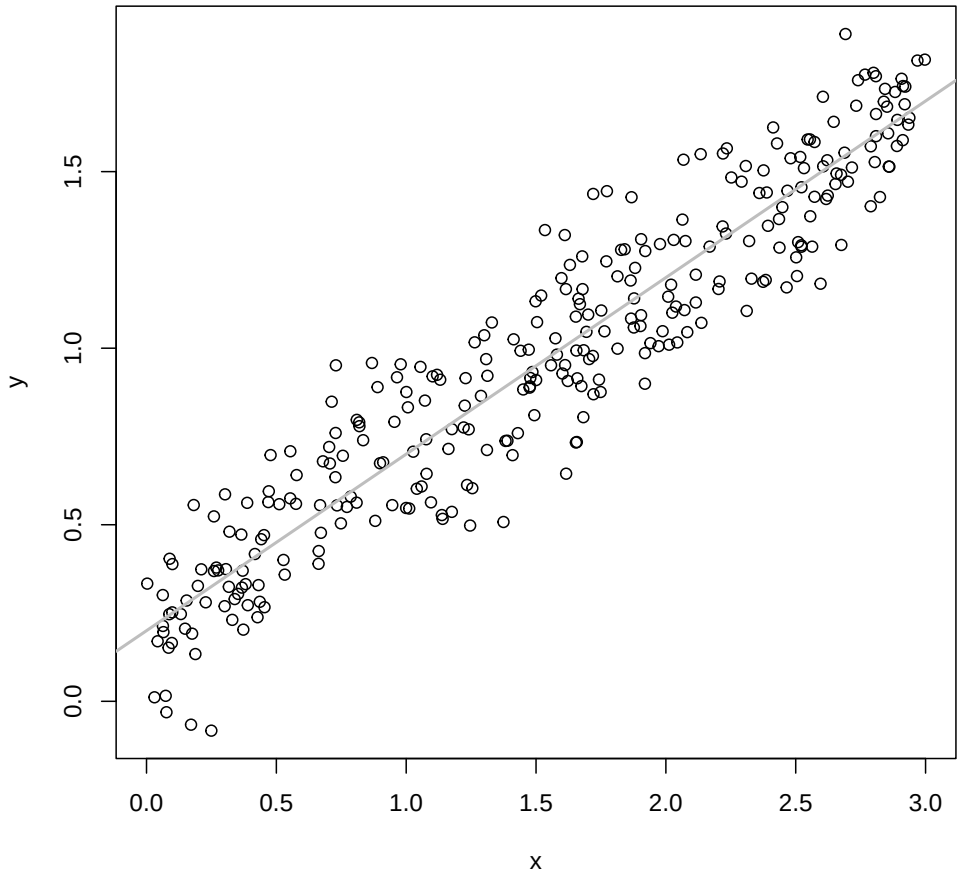
Figure 9.3 Histogram of the distribution of $D = MSE_p - MSE_{np}$ for data simulated from the parametric model. The vertical line marks the observed value. Notice that the mode is positive and the distribution is right-skewed; this is typical.

test should give us a high p-value. So let us stipulate that in reality

$$Y = 0.2 + 0.5x + \eta \tag{9.5}$$

with $\eta \sim \mathcal{N}(0, 0.15^2)$. Figure 9.4 shows data from this, of the same size as before.

Repeating the same exercise as before, we get that $\hat{d} = 0.0013$, together with




```
y2 <- 0.2 + 0.5 * x + rnorm(length(x), 0, 0.15)
y2.frame <- data.frame(x = x, y = y2)
plot(x, y2, xlab = "x", ylab = "y")
abline(0.2, 0.5, col = "grey", lwd = 2)
```

Figure 9.4 Data from the linear model (true regression line in grey).

a slightly different null distribution (Figure 9.5). Now the p-value is 0.14, which it would be quite rash to reject.

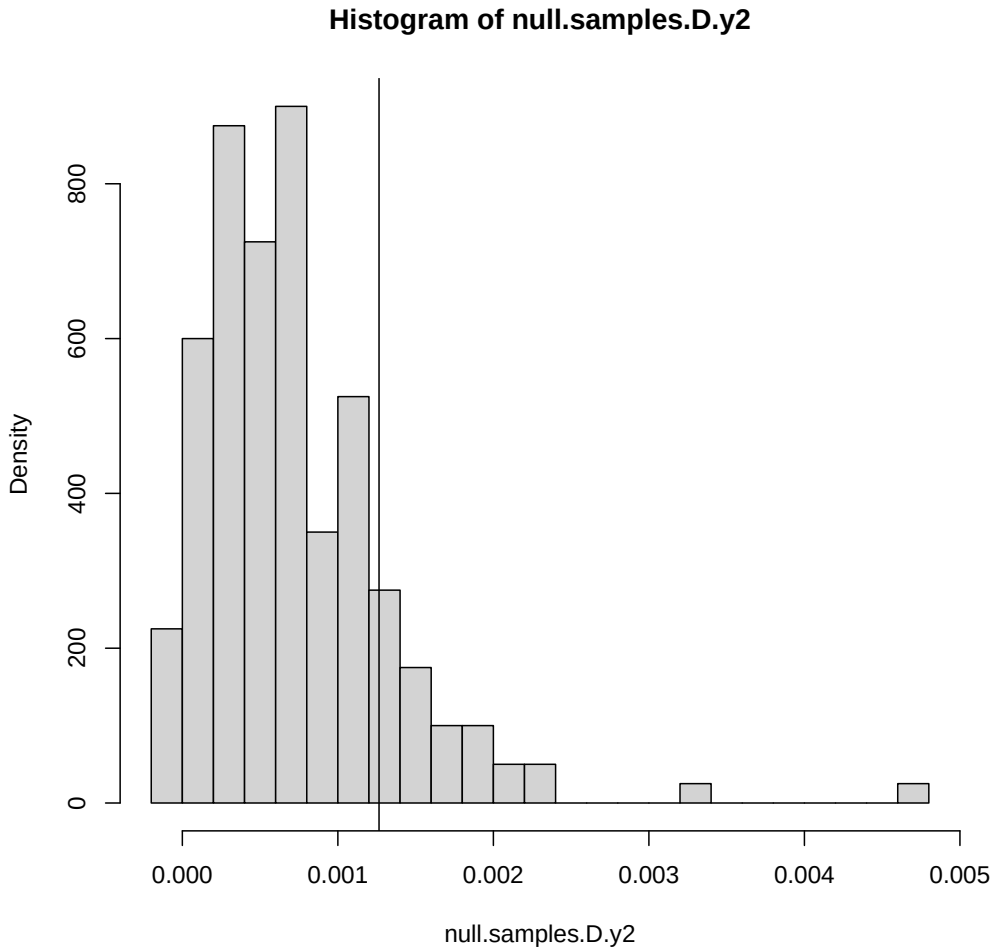


Figure 9.5 As in Figure 9.3, but using the data and fits from Figure 9.4.

9.1.2 Remarks

Other Nonparametric Regressions

There is nothing especially magical about using kernel regression here. Any consistent nonparametric estimator (say, your favorite spline) would work. They may differ somewhat in their answers on particular cases.

Curse of Dimensionality

For multivariate regressions, testing against a fully nonparametric alternative can be very time-consuming, as well as running up against curse-of-dimensionality issues[7]. A compromise is to test the parametric regression against an additive model. Essentially nothing has to change.

Testing $\mathbb{E}[\widehat{\epsilon}|X] = 0$

I mentioned at the beginning of the chapter that one way to test whether the parametric model is correctly specified is to test whether the residuals have expectation zero everywhere. Setting $r(x; m) \equiv \mathbb{E}[Y - m(X)|X = x]$, we know from Chapter **??** that $r(x; \mu) = 0$ everywhere, and that, for any other function m, $r(x; m) \neq 0$ for at least some values of x. Thus, if we take the residuals $\widehat{\epsilon}$ from our parametric model and we smooth them, we get an estimated function $\hat{r}(x)$ that should be converging to 0 everywhere if the parametric model is well-specified. A natural test statistic is therefore some measure of the "size" of $\hat{r}$, such as[8] $\int \hat{r}^2(x)dx$, or $\int \hat{r}^2(x)f(x)dx$ (where $f(x)$ is the pdf of X). (The latter, in particular, can be approximated by $n^{-1}\sum_{i=1}^{n}\hat{r}^2(x_i)$.) Our testing procedure would then amount to (i) finding the residuals by fitting the parametric model, (ii) smoothing the residuals to get $\hat{r}$, (iii) calculating the size of $\hat{r}$, and (iv) simulating to get a distribution for how big $\hat{r}$ should be, under the null hypothesis that the parametric model is right.

An alternative to measuring the size of the expected-residuals function would be to try to predict the residuals. We would compare the MSEs of the "model" that the residuals have conditional expectation 0 everywhere, to the MSE of the model that predicts the residuals by smoothing against X, and proceed much as before[9].

Stabilizing the Sampling Distribution of the Test Statistic

I have just looked at the difference in MSEs. The bootstrap principle being invoked is that the sampling distribution of the test statistic, under the estimated parametric model, should be close to the distribution under the true parameter

[7] This curse manifests itself here as a loss of power in the test. Said another way, because unconstrained non-parametric regression must use a lot of data points just to determine the general shape of the regression function, even more data is needed to tell whether a particular parametric guess is wrong.

[8] If you've taken functional analysis or measure theory, you may recognize these as the (squared) L_2 and $L_2(f)$ norms of the function $\hat{r}$.

[9] Can you write the difference in MSEs for the residuals in terms of either of the measures of the size of $\hat{r}$?

value. As discussed in Chapter 6, sometimes some massaging of the test statistic helps bring these distributions closer. Some modifications to consider:

- Divide the MSE difference by an estimate of the noise σ.
- Divide by an estimate of the noise σ times the difference in degrees of freedom, using the effective degrees of freedom (§1.5.3.2) of the nonparametric regression.
- Use the log of the ratio in MSEs instead of the MSE difference.

Doing a double bootstrap can help you assess whether these are necessary.

9.2 Why Use Parametric Models At All?

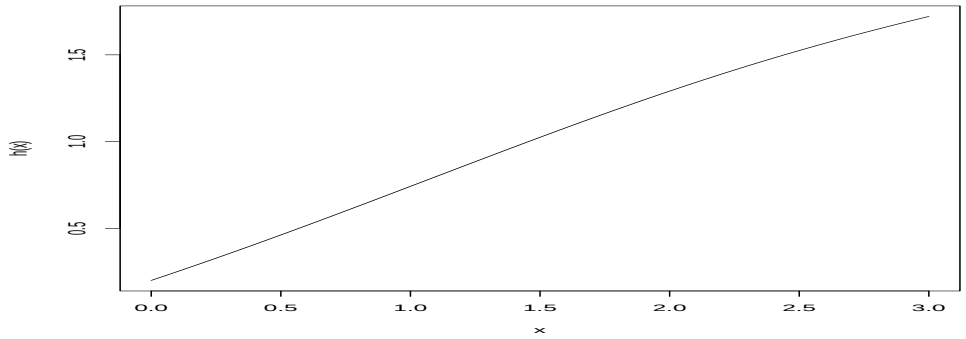
It might seem by this point that there is little point to using parametric models at all. Either our favorite parametric model is right, or it isn't. If it is right, then a consistent nonparametric estimate will eventually approximate it arbitrarily closely. If the parametric model is wrong, it will not self-correct, but the nonparametric estimate will eventually show us that the parametric model doesn't work. Either way, the parametric model seems superfluous.

There are two things wrong with this line of reasoning — two good reasons to use parametric models.

1. One use of statistical models, like regression models, is to connect scientific theories to data. The theories are ideas about the mechanisms generating the data. Sometimes these ideas are precise enough to tell us what the functional form of the regression should be, or even what the distribution of noise terms should be, but still contain unknown parameters. In this case, the parameters themselves are substantively meaningful and interesting — we *don't* just care about prediction.[10]
2. Even if all we care about *is* prediction accuracy, there is still the bias-variance trade-off to consider. Non-parametric smoothers will have larger variance in their predictions, at the same sample size, than correctly-specified parametric models, simply because the former are more flexible. Both models are converging on the true regression function, but the parametric model converges faster, because it searches over a more confined space. In terms of total prediction error, the parametric model's low variance plus vanishing bias beats the nonparametric smoother's larger variance plus vanishing bias. (Remember that this is part of the logic of testing parametric models in the previous section.) In the next section, we will see that this argument can actually be pushed further, to work with not-quite-correctly specified models.

Of course, both of these advantages of parametric models only obtain *if* they are well-specified. If we want to claim those advantages, we need to check the specification.

[10] On the other hand, it is not uncommon for scientists to write down theories positing linear relationships between variables, not because they actually believe that, but because that's the only thing they know how to estimate statistically.




```
h <- function(x) { 0.2 + 0.5*(1+sin(x)/10)*x }
curve(h(x),from=0,to=3)
```

Figure 9.6 Graph of $h(x) = 0.2 + \frac{1}{2}\left(1 + \frac{\sin x}{10}\right) x$ over $[0, 3]$.

9.2.1 Why We Sometimes Want Mis-Specified Parametric Models

Low-dimensional parametric models have potentially high bias (if the real regression curve is very different from what the model posits), but low variance (because there isn't that much to estimate). Non-parametric regression models have low bias (they're flexible) but high variance (they're flexible). If the parametric model is true, it can converge *faster* than the non-parametric one. Even if the parametric model isn't quite true, a small bias plus low variance can sometimes still beat a non-parametric smoother's smaller bias and substantial variance. With enough data the non-parametric smoother will eventually over-take the mis-specified parametric model, but with small samples we might be better off embracing bias.

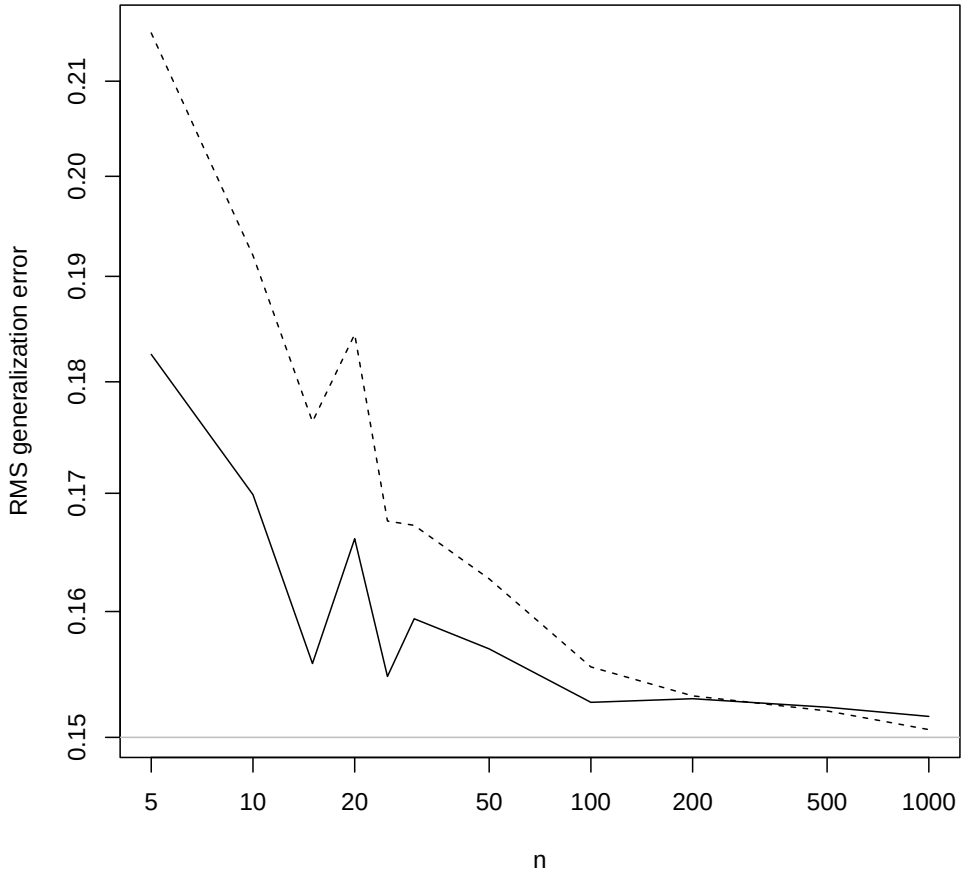
To illustrate, suppose that the true regression function is

$$\mathbb{E}[Y|X = x] = 0.2 + \frac{1}{2}\left(1 + \frac{\sin x}{10}\right) x \tag{9.6}$$

This is very nearly linear over small ranges — say $x \in [0, 3]$ (Figure 9.6).

I will use the fact that I know the true model here to calculate the actual expected generalization error, by averaging over many samples (Example 23).

Figure 9.7 shows that, out to a fairly substantial sample size (≈ 500), the lower bias of the non-parametric regression is systematically beaten by the lower variance of the linear model — though admittedly not by much.




```
sizes <- c(5, 10, 15, 20, 25, 30, 50, 100, 200, 500, 1000)
generalizations <- sapply(sizes, nearly.linear.generalization)
plot(sizes, sqrt(generalizations[1, ]), type = "l", xlab = "n", ylab = "RMS generalization error",
    log = "xy", ylim = range(sqrt(generalizations)))
lines(sizes, sqrt(generalizations[2, ]), lty = "dashed")
abline(h = 0.15, col = "grey")
```

Figure 9.7 Root-mean-square generalization error for linear model (solid line) and kernel smoother (dashed line), fit to the same sample of the indicated size. The true regression curve is as in 9.6, and observations are corrupted by IID Gaussian noise with $\sigma = 0.15$ (grey horizontal line). The cross-over after which the nonparametric regressor has better generalization performance happens shortly before $n = 500$.

```
nearly.linear.out.of.sample = function(n) {
    x <- seq(from = 0, to = 3, length.out = n)
    y <- h(x) + rnorm(n, 0, 0.15)
    data <- data.frame(x = x, y = y)
    y.new <- h(x) + rnorm(n, 0, 0.15)
    sim.lm <- lm(y ~ x, data = data)
    lm.mse <- mean((fitted(sim.lm) - y.new)^2)
    sim.np.bw <- npregbw(y ~ x, data = data)
    sim.np <- npreg(sim.np.bw)
    np.mse <- mean((fitted(sim.np) - y.new)^2)
    mses <- c(lm.mse, np.mse)
    return(mses)
}

nearly.linear.generalization <- function(n, m = 100) {
    raw <- replicate(m, nearly.linear.out.of.sample(n))
    reduced <- rowMeans(raw)
    return(reduced)
}
```

CODE EXAMPLE 23: *Evaluating the out-of-sample error for the nearly-linear problem as a function of n, and evaluting the generalization error by averaging over many samples.*

9.3 Further Reading

This chapter has been on specification testing for regression models, focusing on whether they are correctly specified for the conditional expectation function. I am not aware of any other treatment of this topic at this level, other than the not-wholly-independent Spain *et al.* (2012). If you have somewhat more statistical theory than this book demands, there are very good treatments of related tests in Li and Racine (2007), and of tests based on smoothing residuals in Hart (1997).

Econometrics seems to have more of a tradition of formal specification testing than many other branches of statistics. Godfrey (1988) reviews tests based on looking for parametric extensions of the model, i.e., refinements of the idea of testing whether $\theta_3 = 0$ in Eq. 9.3. White (1994) combines a detailed theory of specification testing within parametric stochastic models, *not* presuming any particular parametric model is correct, with an analysis of when we can and cannot still draw useful inferences from estimates within a mis-specified model. Because of its generality, it, too, is at a higher theoretical level than this book, but is strongly recommend. White was also the co-author of a paper (Hong and White, 1995) presenting a theoretical analysis of the difference-in-MSEs test used in this chapter, albeit for a particular sort of nonparametric regression we've not really touched on.

Appendix F considers some ways of doing specification test for models of distributions, rather than regressions.

10

Moving Beyond Conditional Expectations: Weighted Least Squares, Heteroskedasticity, Local Polynomial Regression

So far, all our estimates have been based on the mean squared error, giving equal importance to all observations, as is generally appropriate when looking at conditional expectations. In this chapter, we'll start to work with giving more or less weight to different observations, through *weighted* least squares. The oldest reason to want to use weighted least squares is to deal with non-constant variance, or heteroskedasticity, by giving more weight to lower-variance observations. This leads us naturally to estimating the conditional variance function, just as we've been estimating conditional expectations. On the other hand, weighted least squares lets us general kernel regression to locally polynomial regression.

10.1 Weighted Least Squares

When we use ordinary least squares to estimate linear regression, we (naturally) minimize the mean squared error:

$$MSE(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \vec{x}_i \cdot \beta)^2 \tag{10.1}$$

The solution is of course

$$\widehat{\beta}_{OLS} = (\mathbf{x}^T\mathbf{x})^{-1}\mathbf{x}^T\mathbf{y} \tag{10.2}$$

We could instead minimize the *weighted* mean squared error,

$$WMSE(\beta, \vec{w}) = \frac{1}{n}\sum_{i=1}^{n} w_i(y_i - \vec{x}_i \cdot \beta)^2 \tag{10.3}$$

This includes ordinary least squares as the special case where all the weights $w_i = 1$. We can solve it by the same kind of linear algebra we used to solve the ordinary linear least squares problem. If we write $\mathbf{w}$ for the matrix with the w_i on the diagonal and zeroes everywhere else, the solution is

$$\widehat{\beta}_{WLS} = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\mathbf{y} \tag{10.4}$$

But why would we want to minimize Eq. 10.3?

1. *Focusing accuracy.* We may care very strongly about predicting the response for certain values of the input — ones we expect to see often again, ones where mistakes are especially costly or embarrassing or painful, etc. — than others.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

If we give the points $\vec{x_i}$ near that region big weights w_i, and points elsewhere smaller weights, the regression will be pulled towards matching the data in that region.

2. *Discounting imprecision.* Ordinary least squares is the maximum likelihood estimate when the ϵ in $Y = \vec{X} \cdot \beta + \epsilon$ is IID Gaussian white noise. This means that the variance of ϵ has to be constant, and we measure the regression curve with the same precision elsewhere. This situation, of constant noise variance, is called **homoskedasticity**. Often however the magnitude of the noise is not constant, and the data are **heteroskedastic**.

   When we have heteroskedasticity, even if each noise term is still Gaussian, ordinary least squares is no longer the maximum likelihood estimate, and so no longer efficient. If however we know the noise variance σ_i^2 at each measurement i, and set $w_i = 1/\sigma_i^2$, we get the heteroskedastic MLE, and recover efficiency. (See below.)

   To say the same thing slightly differently, there's just no way that we can estimate the regression function as accurately where the noise is large as we can where the noise is small. Trying to give equal attention to all parts of the input space is a waste of time; we should be more concerned about fitting well where the noise is small, and expect to fit poorly where the noise is big.

3. *Sampling bias.* In many situations, our data comes from a survey, and some members of the population may be more likely to be included in the sample than others. When this happens, the sample is a biased representation of the population. If we want to draw inferences about the population, it can help to give more weight to the kinds of data points which we've under-sampled, and less to those which were over-sampled. In fact, typically the weight put on data point i would be inversely proportional to the probability of i being included in the sample (exercise 10.1). Strictly speaking, if we are willing to believe that linear model is exactly correct, that there are no omitted variables, and that the inclusion probabilities p_i do not vary with y_i, then this sort of survey weighting is redundant (DuMouchel and Duncan, 1983). When those assumptions are not met — when there're non-linearities, omitted variables, or "selection on the dependent variable" — survey weighting is advisable, if we know the inclusion probabilities fairly well.

   The same trick works under the same conditions when we deal with "covariate shift", a change in the distribution of X. If the old probability density function was $p(x)$ and the new one is $q(x)$, the weight we'd want to use is $w_i = q(x_i)/p(x_i)$ (Quiñonero-Candela *et al.*, 2009). This can involve estimating both densities, or their ratio (chapter 14).

4. *Doing something else.* There are a number of other optimization problems which can be transformed into, or approximated by, weighted least squares. The most important of these arises from generalized linear models, where the mean response is some nonlinear function of a linear predictor; we will look at them in Chapters 11 and 12.

In the first case, we decide on the weights to reflect our priorities. In the

third case, the weights come from the optimization problem we'd really rather be solving. What about the second case, of heteroskedasticity?

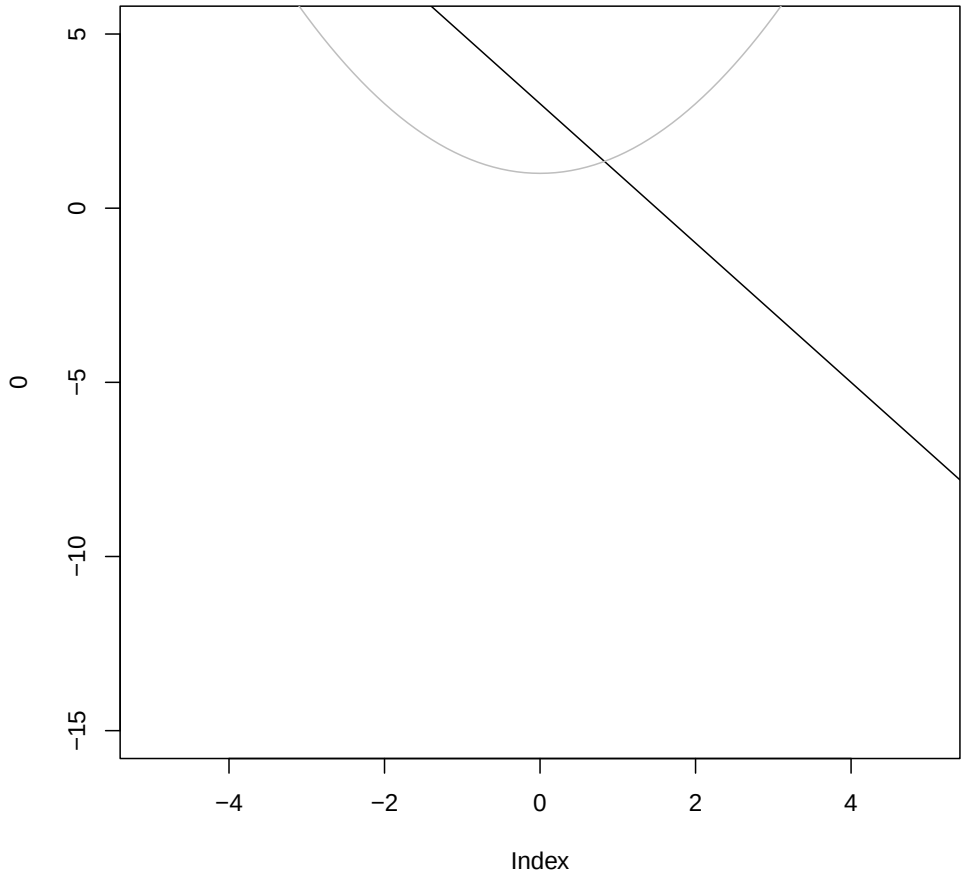
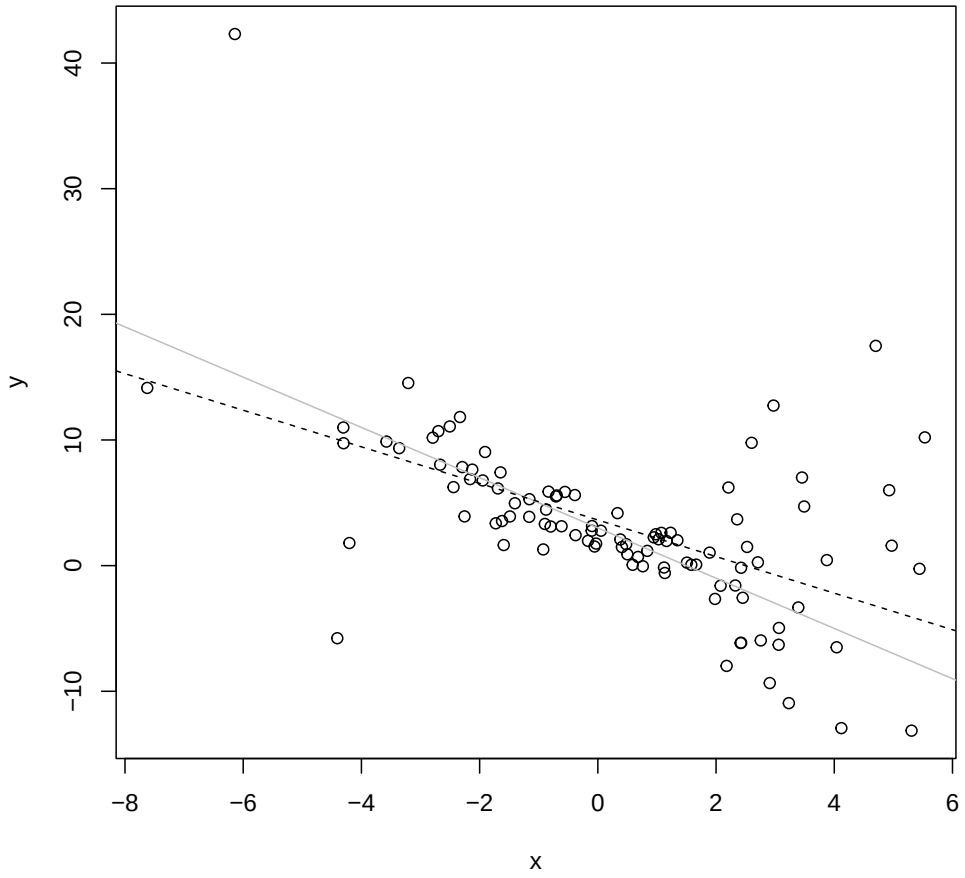


Figure 10.1 Black line: Linear response function ($y = 3 - 2x$). Grey curve: standard deviation as a function of x ($\sigma(x) = 1 + x^2/2$). (Code deliberately omitted; can you reproduce this figure?)

10.2 Heteroskedasticity

Suppose the noise variance is itself variable. For example, the figure shows a simple linear relationship between the input X and the response Y, but also a nonlinear relationship between X and $\mathbb{V}[Y]$.

In this particular case, the ordinary least squares estimate of the regression line is $3.63 - -1.46x$, with R reporting standard errors in the coefficients of ± 0.59 and 0.22, respectively. Those are however calculated under the assumption that the noise is homoskedastic, which it isn't. And in fact we can see, pretty much,

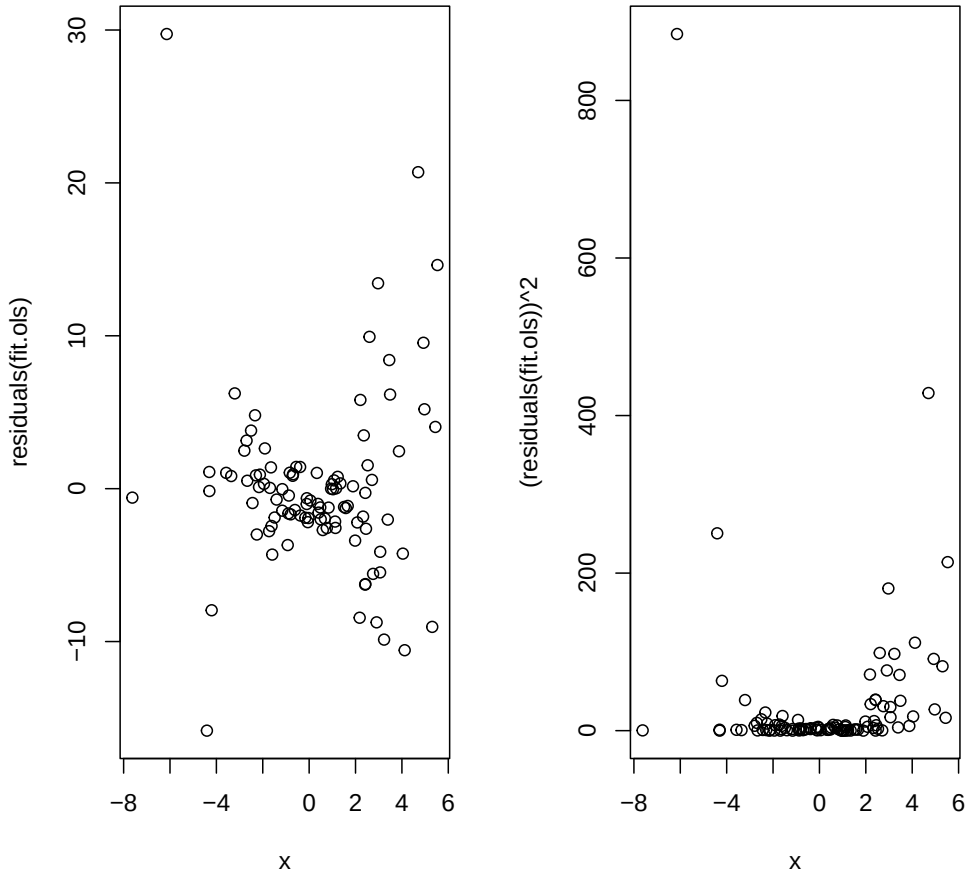



```
plot(x, y)
abline(a = 3, b = -2, col = "grey")
fit.ols = lm(y ~ x)
abline(fit.ols, lty = "dashed")
```

Figure 10.2 Scatter-plot of $n = 100$ data points from the above model. (Here X is Gaussian with mean 0 and variance 9.) Grey: True regression line. Dashed: ordinary least squares regression line.

that there is heteroskedasticity — if looking at the scatter-plot didn't convince us, we could always plot the residuals against x, which we should do anyway.

To see whether that makes a difference, let's re-do this many times with different draws from the same model (Example 24).




```
par(mfrow = c(1, 2))
plot(x, residuals(fit.ols))
plot(x, (residuals(fit.ols))^2)
par(mfrow = c(1, 1))
```

Figure 10.3 Residuals (left) and squared residuals (right) of the ordinary least squares regression as a function of x. Note the much greater range of the residuals at large absolute values of x than towards the center; this changing dispersion is a sign of heteroskedasticity.

Running `ols.heterosked.error.stats(1e4)` produces 10^4 random simulated data sets, which all have the same x values as the first one, but different values of y, generated however from the same model. It then uses those samples to get

```
ols.heterosked.example = function(n) {
    y = 3 - 2 * x + rnorm(n, 0, sapply(x, function(x) {
        1 + 0.5 * x^2
    }))
    fit.ols = lm(y ~ x)
    return(fit.ols$coefficients - c(3, -2))
}

ols.heterosked.error.stats = function(n, m = 10000) {
    ols.errors.raw = t(replicate(m, ols.heterosked.example(n)))
    intercept.se = sd(ols.errors.raw[, "(Intercept)"])
    slope.se = sd(ols.errors.raw[, "x"])
    return(c(intercept.se = intercept.se, slope.se = slope.se))
}
```

CODE EXAMPLE 24: *Functions to generate heteroskedastic data and fit OLS regression to it, and to collect error statistics on the results.*

the standard error of the ordinary least squares estimates. (Bias remains a non-issue.) What we find is the standard error of the intercept is only a little inflated (simulation value of 0.7 versus official value of 0.59), but the standard error of the slope is much larger than what R reports, 0.52 versus 0.22. Since the intercept is fixed by the need to make the regression line go through the center of the data (Chapter 2), the real issue here is that our estimate of the slope is much less precise than ordinary least squares makes it out to be. Our estimate is still consistent, but not as good as it was when things were homoskedastic. Can we get back some of that efficiency?

10.2.1 Weighted Least Squares as a Solution to Heteroskedasticity

Suppose we visit the Oracle of Regression (Figure 10.4), who tells us that the noise has a standard deviation that goes as $1 + x^2/2$. We can then use this to improve our regression, by solving the weighted least squares problem rather than ordinary least squares (Figure 10.5).

This not only looks better, it is better: the estimated line is now $2.95 - 1.53x$, with reported standard errors of 0.32 and 0.2. This checks check out with simulation (Example 25): the standard errors from the simulation are 0.23 for the intercept and 0.23 for the slope, so R's internal calculations are working very well.

Why does putting these weights into WLS improve things?

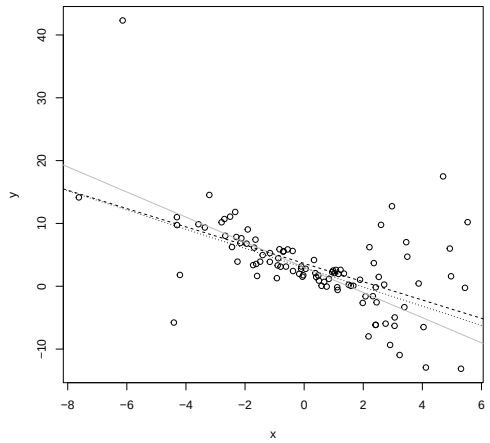


Figure 10.4 Statistician (right) consulting the Oracle of Regression (left) about the proper weights to use to overcome heteroskedasticity. (Image from http://en.wikipedia.org/wiki/Image:Pythia1.jpg.)

```
wls.heterosked.example = function(n) {
    y = 3 - 2 * x + rnorm(n, 0, sapply(x, function(x) {
        1 + 0.5 * x^2
    }))
    fit.wls = lm(y ~ x, weights = 1/(1 + 0.5 * x^2))
    return(fit.wls$coefficients - c(3, -2))
}

wls.heterosked.error.stats = function(n, m = 10000) {
    wls.errors.raw = t(replicate(m, wls.heterosked.example(n)))
    intercept.se = sd(wls.errors.raw[, "(Intercept)"])
    slope.se = sd(wls.errors.raw[, "x"])
    return(c(intercept.se = intercept.se, slope.se = slope.se))
}
```

Code Example 25: *Linear regression of heteroskedastic data, using weighted least-squared regression.*




```
plot(x, y)
abline(a = 3, b = -2, col = "grey")
fit.ols = lm(y ~ x)
abline(fit.ols, lty = "dashed")
fit.wls = lm(y ~ x, weights = 1/(1 + 0.5 * x^2))
abline(fit.wls, lty = "dotted")
```

Figure 10.5 Figure 10.2, plus the weighted least squares regression line (dotted).

10.2.2 Some Explanations for Weighted Least Squares

Qualitatively, the reason WLS with inverse variance weights works is the following. OLS tries equally hard to match observations at each data point.[1] Weighted least squares, naturally enough, tries harder to match observations where the weights are big, and less hard to match them where the weights are small. But each y_i contains not only the true regression function $\mu(x_i)$ but also some noise ϵ_i. The noise terms have large magnitudes where the variance is large. So we should want to have small weights where the noise variance is large, because there the data tends to be far from the true regression. Conversely, we should put big weights where the noise variance is small, and the data points are close to the true regression.

The qualitative reasoning in the last paragraph doesn't explain why the weights should be inversely proportional to the variances, $w_i \propto 1/\sigma^2_{x_i}$ — why not $w_i \propto 1/\sigma_{x_i}$, for instance? Seeing why those are the right weights requires investigating how well different, indeed *arbitrary*, choices of weights would work.

Look at the equation for the WLS estimates again:

$$\widehat{\beta}_{WLS} = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}\mathbf{y} \tag{10.5}$$

$$= \mathbf{h}(\mathbf{w})\mathbf{y} \tag{10.6}$$

defining the matrix $\mathbf{h}(\mathbf{w}) = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w}$ for brevity. (The notation reminds us that everything depends on the weights in $\mathbf{w}$.) Imagine holding $\mathbf{x}$ constant, but repeating the experiment multiple times, so that we get noisy values of $\mathbf{y}$. In each experiment, $Y_i = \vec{x}_i \cdot \beta + \epsilon_i$, where $\mathbb{E}[\epsilon_i] = 0$ and $\mathbb{V}[\epsilon_i] = \sigma^2_{x_i}$. So

$$\widehat{\beta}_{WLS} = \mathbf{h}(\mathbf{w})\mathbf{x}\beta + \mathbf{h}(\mathbf{w})\epsilon \tag{10.7}$$

$$= \beta + \mathbf{h}(\mathbf{w})\epsilon \tag{10.8}$$

Since $\mathbb{E}[\epsilon] = 0$, the WLS estimator is unbiased:

$$\mathbb{E}\left[\widehat{\beta}_{WLS}\right] = \beta \tag{10.9}$$

In fact, for the jth coefficient,

$$\widehat{\beta}_j = \beta_j + [\mathbf{h}(\mathbf{w})\epsilon]_j \tag{10.10}$$

$$= \beta_j + \sum_{i=1}^{n} h_{ji}(\mathbf{w})\epsilon_i \tag{10.11}$$

Since the WLS estimate is unbiased, it's natural to want it to also have a small variance, and

$$\mathbb{V}\left[\widehat{\beta}_j\right] = \sum_{i=1}^{n} h_{ji}(\mathbf{w})\sigma^2_{x_i} \tag{10.12}$$

It can be shown — the result is called the **Gauss-Markov theorem** — that

[1] Less anthropomorphically, the objective function in Eq. 10.1 has the same derivative with respect to the squared error at each point, $\frac{\partial MSE}{\partial(y_i - \vec{x}_i\cdot\beta)^2} = \frac{1}{n}$.

picking weights to minimize the variance in the WLS estimate has the unique solution $w_i = 1/\sigma^2_{x_i}$. It does not require us to assume the noise is Gaussian[2], but the proof is a bit tricky, so I will confine it to §10.2.2.1 below.

A less general but easier-to-grasp result comes from adding the assumption that the noise around the regression line is Gaussian — that

$$Y = \vec{x} \cdot \beta + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2_x) \tag{10.13}$$

The log-likelihood is then (Exercise 10.2)

$$-\frac{n}{2}\ln 2\pi - \frac{1}{2}\sum_{i=1}^{n}\log \sigma^2_{x_i} - \frac{1}{2}\sum_{i=1}^{n}\frac{(y_i - \vec{x}_i \cdot \beta)^2}{\sigma^2_{x_i}} \tag{10.14}$$

If we maximize this with respect to β, everything except the final sum is irrelevant, and so we minimize

$$\sum_{i=1}^{n}\frac{(y_i - \vec{x}_i \cdot \beta)^2}{\sigma^2_{x_i}} \tag{10.15}$$

which is just weighted least squares with $w_i = 1/\sigma^2_{x_i}$. So, if the probabilistic assumption holds, WLS is the efficient maximum likelihood estimator.

[2] Despite the first part of the name! Gauss himself was much less committed to assuming Gaussian distributions than many later statisticians.

10.2.2.1 Proof of the Gauss-Markov Theorem[3]

We want to prove that, when we are doing weighted least squares for linear regression, the best choice of weights $w_i = 1/\sigma^2_{x_i}$. We saw that that WLS is unbiased (Eq. 10.9), so "best" here means minimizing the variance. We have also already seen (Eq. 10.6) that

$$\widehat{\beta}_{WLS} = \mathbf{h}(\mathbf{w})\mathbf{y} \tag{10.16}$$

where the matrix $\mathbf{h}(\mathbf{w})$ is

$$\mathbf{h}(\mathbf{w}) = (\mathbf{x}^T\mathbf{w}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w} \tag{10.17}$$

It would be natural to try to write out the variance as a function of the weights w, set the derivative equal to zero, and solve. This is tricky, partly because we need to make sure that all the weights are positive and add up to one, but mostly because of the matrix inversion in the definition of $\mathbf{h}(\mathbf{w})$. A slightly less direct approach is actually much cleaner.

Write $\mathbf{w_0}$ for the inverse-variance weight matrix, and $\mathbf{h_0}$ for the hat matrix we get with those weights. Then for any other choice of weights, we have $\mathbf{h}(\mathbf{w}) = \mathbf{h_0} + \mathbf{c}$. ($\mathbf{c}$ is implicitly a function of the weights, but let's suppress that in the notation for brevity.) Since we know all WLS estimates are unbiased, we must have

$$(\mathbf{h_0} + \mathbf{c})\mathbf{x}\beta = \beta \tag{10.18}$$

but using the inverse-variance weights is a particular WLS estimate so

$$\mathbf{h_0}\mathbf{x}\beta = \beta \tag{10.19}$$

and so we can deduce that

$$\mathbf{c}\mathbf{x} = 0 \tag{10.20}$$

from unbiasedness.

Now consider the covariance matrix of the estimates, $\mathbb{V}\left[\tilde{\beta}\right]$. This will be $\mathbb{V}\left[(\mathbf{h_0} + \mathbf{c})\mathbf{Y}\right]$,

[3] You can skip this section, without loss of continuity.

which we can expand:

$$\begin{aligned}
\mathbb{V}\left[\tilde{\beta}\right] &= \mathbb{V}\left[(\mathbf{h_0}+\mathbf{c})\mathbf{Y}\right] && (10.21)\\
&= (\mathbf{h_0}+\mathbf{c})\mathbb{V}\left[Y\right](\mathbf{h_0}+\mathbf{c})^T && (10.22)\\
&= (\mathbf{h_0}+\mathbf{c})\mathbf{w_0}^{-1}(\mathbf{h_0}+\mathbf{c})^T && (10.23)\\
&= \mathbf{h_0}\mathbf{w_0}^{-1}\mathbf{h_0}^T + \mathbf{c}\mathbf{w_0}^{-1}\mathbf{h_0}^T + \mathbf{h_0}\mathbf{w_0}^{-1}\mathbf{c}^T + \mathbf{c}\mathbf{w_0}^{-1}\mathbf{c}^T && (10.24)\\
&= (\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w_0}\mathbf{w_0}^{-1}\mathbf{w_0}\mathbf{x}(\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1} && (10.25)\\
&\quad +\mathbf{c}\mathbf{w_0}^{-1}\mathbf{w_0}\mathbf{x}(\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1}\\
&\quad +(\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w_0}\mathbf{w_0}^{-1}\mathbf{c}^T\\
&\quad +\mathbf{c}\mathbf{w_0}^{-1}\mathbf{c}^T\\
&= (\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{w_0}\mathbf{x}(\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1} && (10.26)\\
&\quad +\mathbf{c}\mathbf{x}(\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1} + (\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1}\mathbf{x}^T\mathbf{c}^T\\
&\quad +\mathbf{c}\mathbf{w_0}^{-1}\mathbf{c}^T\\
&= (\mathbf{x}^T\mathbf{w_0}\mathbf{x})^{-1} + \mathbf{c}\mathbf{w_0}^{-1}\mathbf{c}^T && (10.27)
\end{aligned}$$

where in the last step we use the fact that $\mathbf{cx} = 0$ (and so $\mathbf{x}^T\mathbf{c}^T = 0^T = 0$). Since $\mathbf{c}\mathbf{w_0}^{-1}\mathbf{c}^T \geq 0$, because $\mathbf{w_0}$ is a positive-definite matrix, we see that the variance is minimized by setting $\mathbf{c} = 0$, and using the inverse variance weights.

Notes:

1. If all the variances are equal, then we've proved the optimality of OLS.
2. The proof actually works when comparing the inverse-variance weights to any other linear, unbiased estimator; WLS with different weights is just a special case.
3. We can write the WLS problem as that of minimizing $(\mathbf{y}-\mathbf{x}\beta)^T\mathbf{w}(\mathbf{y}-\mathbf{x}\beta)$. If we allow $\mathbf{w}$ to be a non-diagonal, but still positive-definite, matrix, then we have the **generalized least squares** problem. This is appropriate when there are correlations between the noise terms at different observations, i.e., when $\mathrm{Cov}\left[\epsilon_i, \epsilon_j\right] \neq 0$ even though $i \neq j$. In this case, the proof is easily adapted to show that the optimal weight matrix $\mathbf{w}$ is the inverse of the noise covariance matrix. (This is why I wrote everything as a function of $\mathbf{w}$.)



Figure 10.6 The Oracle may be out (left), or too creepy to go visit (right). What then? (Left, the sacred oak of the Oracle of Dodona, copyright 2006 by Flickr user "essayen", `http://flickr.com/photos/essayen/245236125/`; right, the entrace to the cave of the Sibyl of Cumæ, copyright 2005 by Flickr user "pverdicchio", `http://flickr.com/photos/occhio/17923096/`. Both used under Creative Commons license.) [[ATTN: Both are only licensed for non-commercial use, so find substitutes OR obtain rights for the for-money version of the book]]

10.2.3 Finding the Variance and Weights

All of this was possible because the Oracle told us what the variance function was. What do we do when the Oracle is not available (Figure 10.6)?

Sometimes we can work things out for ourselves, without needing an oracle.

- We know, empirically, the precision of our measurement of the response variable — we know how precise our instruments are, or the response is really an average of several measurements with known standard deviations, etc.
- We know how the noise in the response must depend on the input variables. For example, when taking polls or surveys, the variance of the proportions we find should be inversely proportional to the sample size. So we can make the weights proportional to the sample size.

Both of these outs rely on kinds of background knowledge which are easier to get in the natural or even the social sciences than in many industrial applications. However, there are approaches for other situations which try to use the observed residuals to get estimates of the heteroskedasticity; this is the topic of the next section.

10.3 Estimating Conditional Variance Functions

Remember that there are two equivalent ways of defining the variance:

$$\mathbb{V}[X] = \mathbb{E}\left[X^2\right] - \left(\mathbb{E}[X]\right)^2 = \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \tag{10.28}$$

The latter is more useful for us when it comes to estimating variance functions. We have already figured out how to estimate means — that's what all this previous work on smoothing and regression is for — and the deviation of a random variable from its mean shows up as a residual.

There are two generic ways to estimate conditional variances, which differ slightly in how they use non-parametric smoothing. We can call these the **squared residuals method** and the **log squared residuals method**. Here is how the first one goes.

1. Estimate $\mu(x)$ with your favorite regression method, getting $\widehat{\mu}(x)$.
2. Construct the **squared residuals**, $u_i = (y_i - \widehat{\mu}(x_i))^2$.
3. Use your favorite *non-parametric* method to estimate the conditional mean of the u_i, call it $\widehat{q}(x)$.
4. Predict the variance using $\widehat{\sigma}^2_x = \widehat{q}(x)$.

The log-squared residuals method goes very similarly.

1. Estimate $\mu(x)$ with your favorite regression method, getting $\widehat{\mu}(x)$.
2. Construct the **log squared residuals**, $z_i = \log (y_i - \widehat{\mu}(x_i))^2$.
3. Use your favorite *non-parametric* method to estimate the conditional mean of the z_i, call it $\hat{s}(x)$.
4. Predict the variance using $\widehat{\sigma}^2_x = \exp \widehat{s}(x)$.

The quantity $y_i - \widehat{\mu}(x_i)$ is the ith residual. If $\widehat{\mu} \approx \mu$, then the residuals should have mean zero. Consequently the variance of the residuals (which is what we want) should equal the expected squared residual. So squaring the residuals makes sense, and the first method just smoothes these values to get at their expectations.

What about the second method — why the log? Basically, this is a convenience — squares are necessarily non-negative numbers, but lots of regression methods don't easily include constraints like that, and we really don't want to predict negative variances.[4] Taking the log gives us an unbounded range for the regression.

Strictly speaking, we don't need to use non-parametric smoothing for either method. If we had a parametric model for σ^2_x, we could just fit the parametric model to the squared residuals (or their logs). But even if you think you know what the variance function should look like it, why not check it?

We came to estimating the variance function because of wanting to do weighted least squares, but these methods can be used more generally. It's often important

[4] Occasionally people do things like claiming that gene differences explains more than 100% of the variance in some psychological trait, and so environment and up-bringing contribute negative variance. Some of them — like Alford *et al.* (2005) — say this with a straight face.

to understand variance in its own right, and this is a general method for estimating it. Our estimate of the variance function depends on first having a good estimate of the regression function

10.3.1 Iterative Refinement of Mean and Variance: An Example

The estimate $\widehat{\sigma}_x^2$ depends on the initial estimate of the regression function $\widehat{\mu}$. But, as we saw when we looked at weighted least squares, taking heteroskedasticity into account can change our estimates of the regression function. This suggests an iterative approach, where we alternate between estimating the regression function and the variance function, using each to improve the other. That is, we take either method above, and then, once we have estimated the variance function $\widehat{\sigma}_x^2$, we re-estimate $\widehat{\mu}$ using weighted least squares, with weights inversely proportional to our estimated variance. Since this will generally change our estimated regression, it will change the residuals as well. Once the residuals have changed, we should re-estimate the variance function. We keep going around this cycle until the change in the regression function becomes so small that we don't care about further modifications. It's hard to give a strict guarantee, but *usually* this sort of iterative improvement will converge.

Let's apply this idea to our example. Figure 10.3b already plotted the residuals from OLS. Figure 10.7 shows those squared residuals again, along with the true variance function and the estimated variance function.

The OLS estimate of the regression line is not especially good ($\widehat{\beta}_0 = 3.63$ versus $\beta_0 = 3$, $\widehat{\beta}_1 = -1.46$ versus $\beta_1 = -2$), so the residuals are systematically off, but it's clear from the figure that kernel smoothing of the squared residuals is picking up on the heteroskedasticity, and getting a pretty reasonable picture of the variance function.

Now we use the estimated variance function to re-estimate the regression line, with weighted least squares.

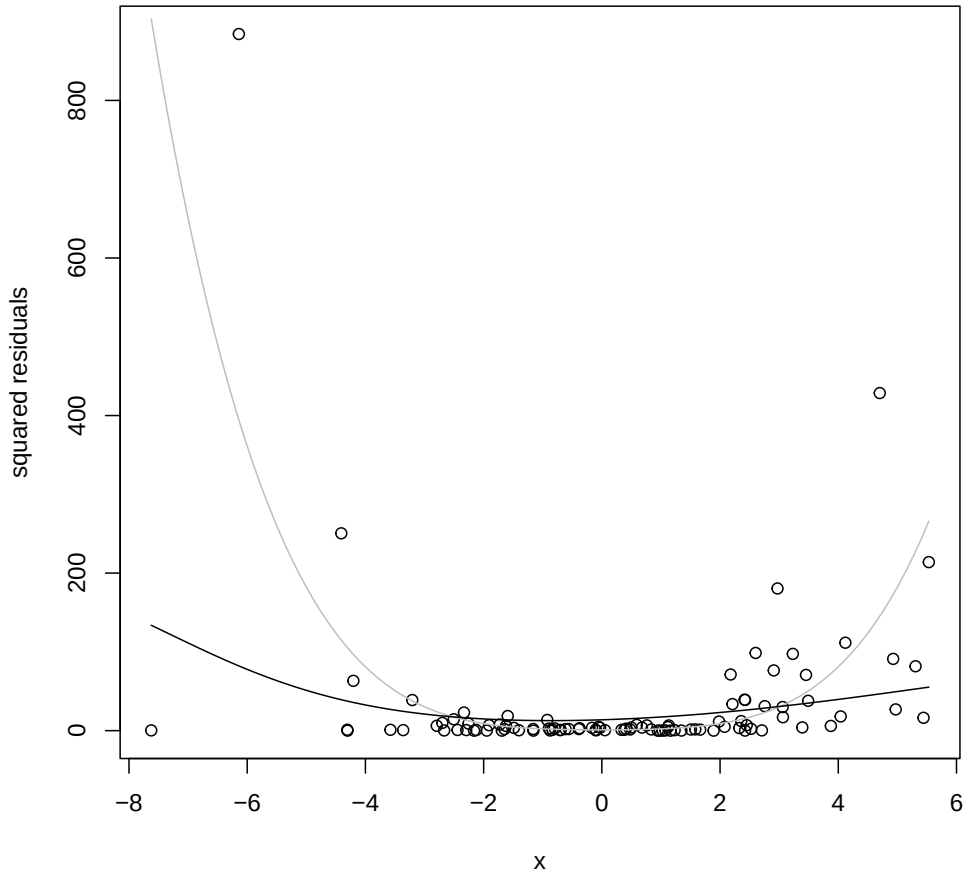
```
fit.wls1 <- lm(y ~ x, weights = 1/fitted(var1))
coefficients(fit.wls1)
## (Intercept)           x
##    3.212162   -1.476736
var2 <- npreg(residuals(fit.wls1)^2 ~ x)
```

The slope has changed substantially, and in the right direction (Figure 10.8a). The residuals have also changed (Figure 10.8b), and the new variance function is closer to the truth than the old one.

Since we have a new variance function, we can re-weight the data points and re-estimate the regression:

```
fit.wls2 <- lm(y ~ x, weights = 1/fitted(var2))
coefficients(fit.wls2)
## (Intercept)           x
##    3.203988   -1.480743
var3 <- npreg(residuals(fit.wls2)^2 ~ x)
```

Since we know that the true coefficients are 3 and -2, we know that this is moving in the right direction. If I hadn't told you what they were, you could still observe that the difference in coefficients between `fit.wls1` and `fit.wls2` is smaller than that between `fit.ols` and `fit.wls1`, which is a sign that this is converging.




```
plot(x, residuals(fit.ols)^2, ylab = "squared residuals")
curve((1 + x^2/2)^2, col = "grey", add = TRUE)
require(np)
var1 <- npreg(residuals(fit.ols)^2 ~ x)
grid.x <- seq(from = min(x), to = max(x), length.out = 300)
lines(grid.x, predict(var1, exdat = grid.x))
```

Figure 10.7 Points: actual squared residuals from the OLS line. Grey curve: true variance function, $\sigma_x^2 = (1 + x^2/2)^2$. Black curve: kernel smoothing of the squared residuals, using `npreg`.

I will spare you the plot of the new regression and of the new residuals. Let's iterate a few more times:

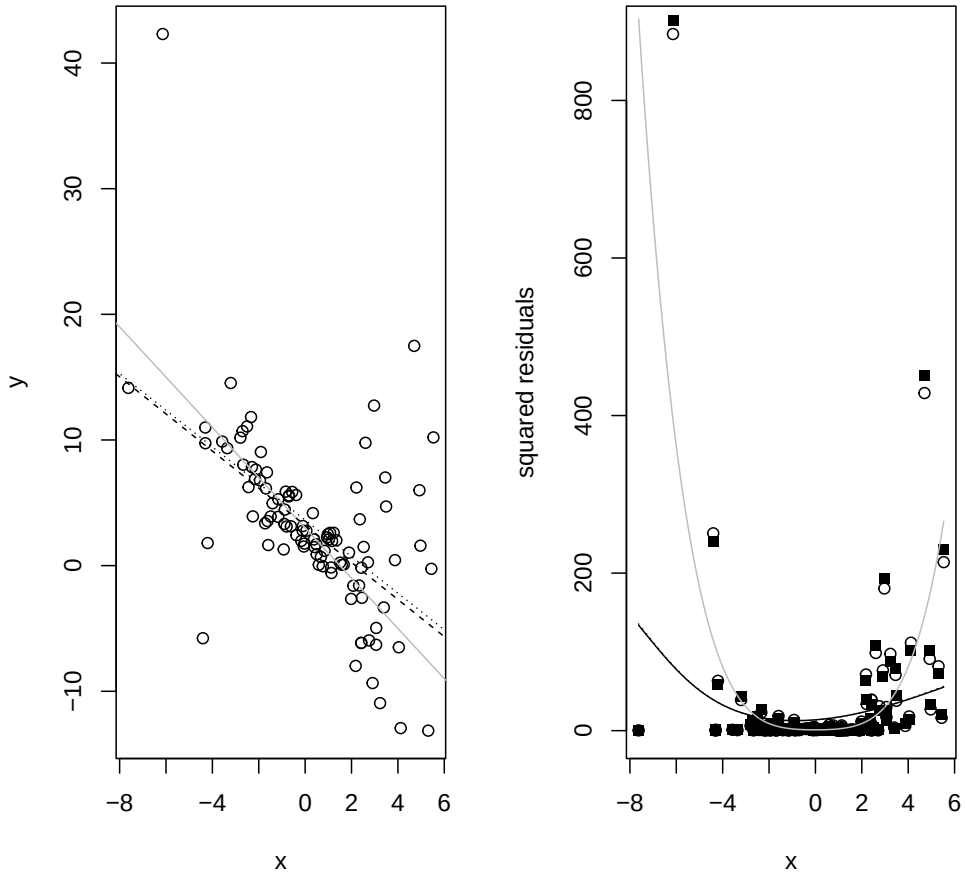


Figure 10.8 Left: As in Figure 10.2, but with the addition of the weighted least squares regression line (dotted), using the estimated variance from Figure 10.7 for weights. Right: As in Figure 10.7, but with the addition of the residuals from the WLS regression (black squares), and the new estimated variance function (dotted curve).

```
fit.wls3 <- lm(y ~ x, weights = 1/fitted(var3))
coefficients(fit.wls3)
## (Intercept)           x
##    3.203520   -1.481161
var4 <- npreg(residuals(fit.wls3)^2 ~ x)
fit.wls4 <- lm(y ~ x, weights = 1/fitted(var4))
coefficients(fit.wls4)
## (Intercept)           x
##    3.203475   -1.481204
```

By now, the coefficients of the regression are changing in the fourth significant digit, and we only have 100 data points, so the imprecision from a limited sample surely swamps the changes we're making, and we might as well stop.

Manually going back and forth between estimating the regression function and estimating the variance function is tedious. We could automate it with a function, which would look something like this:

```
iterative.wls <- function(x, y, tol = 0.01, max.iter = 100) {
    iteration <- 1
    old.coefs <- NA
    regression <- lm(y ~ x)
    coefs <- coefficients(regression)
    while (is.na(old.coefs) || ((max(coefs - old.coefs) > tol) && (iteration < max.iter))) {
        variance <- npreg(residuals(regression)^2 ~ x)
        old.coefs <- coefs
        iteration <- iteration + 1
        regression <- lm(y ~ x, weights = 1/fitted(variance))
        coefs <- coefficients(regression)
    }
    return(list(regression = regression, variance = variance, iterations = iteration))
}
```

This starts by doing an unweighted linear regression, and then alternates between WLS for the getting the regression and kernel smoothing for getting the variance. It stops when no parameter of the regression changes by more than `tol`, or when it's gone around the cycle `max.iter` times.[5] This code is a bit too inflexible to be really "industrial strength" (what if we wanted to use a data frame, or a more complex regression formula?), but shows the core idea.

[5] The condition in the `while` loop is a bit complicated, to ensure that the loop is executed at least once. Some languages have an `until` control structure which would simplify this.

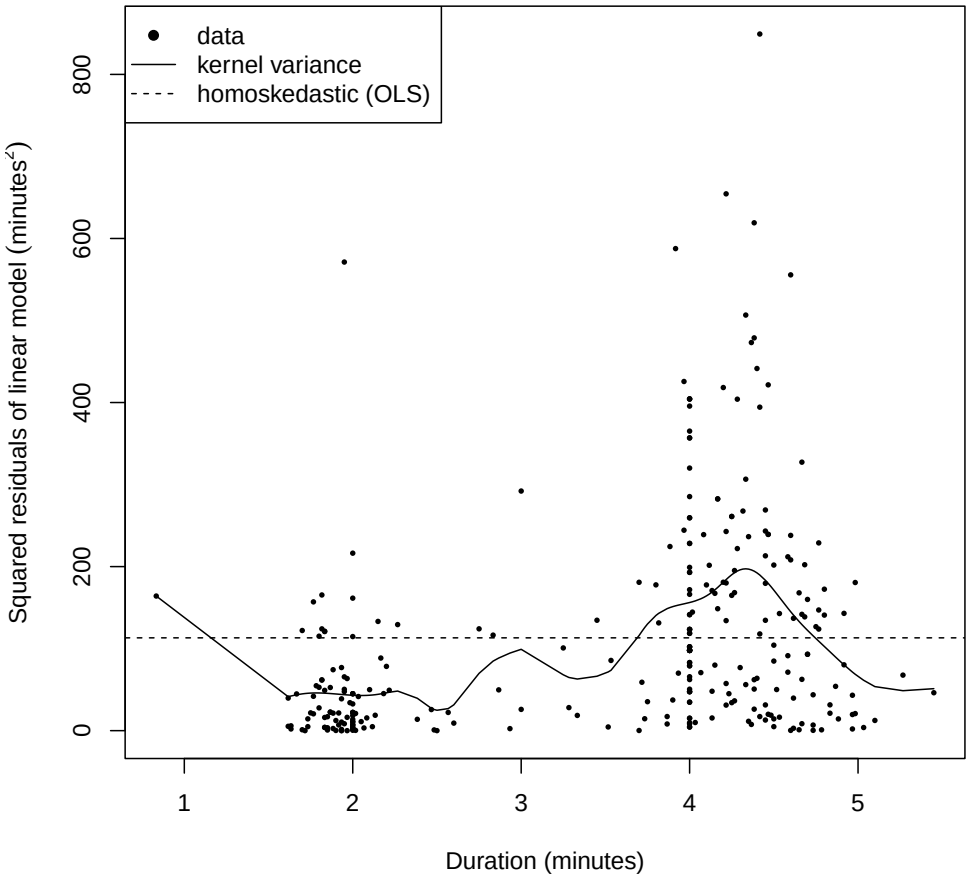
10.3.2 Real Data Example: Old Heteroskedastic

§5.4.2 introduced the `geyser` data set, which is about predicting the waiting time between consecutive eruptions of the "Old Faithful" geyser at Yellowstone National Park from the duration of the latest eruption. Our exploration there showed that a simple linear model (of the kind often fit to this data in textbooks and elementary classes) is not very good, and raised the suspicion that one important problem was heteroskedasticity. Let's follow up on that, building on the computational work done in that section.

The estimated variance function `geyser.var` does not look particularly flat, but it comes from applying a fairly complicated procedure (kernel smoothing with data-driven bandwidth selection) to a fairly limited amount of data (299 observations). Maybe that's the amount of wiggliness we should *expect* to see due to finite-sample fluctuations? To rule this out, we can make surrogate data from the homoskedastic model, treat it the same way as the real data, and plot the resulting variance functions (Figure 10.10). The conditional variance functions estimated from the homoskedastic model are flat or gently varying, with much less range than what's seen in the data.

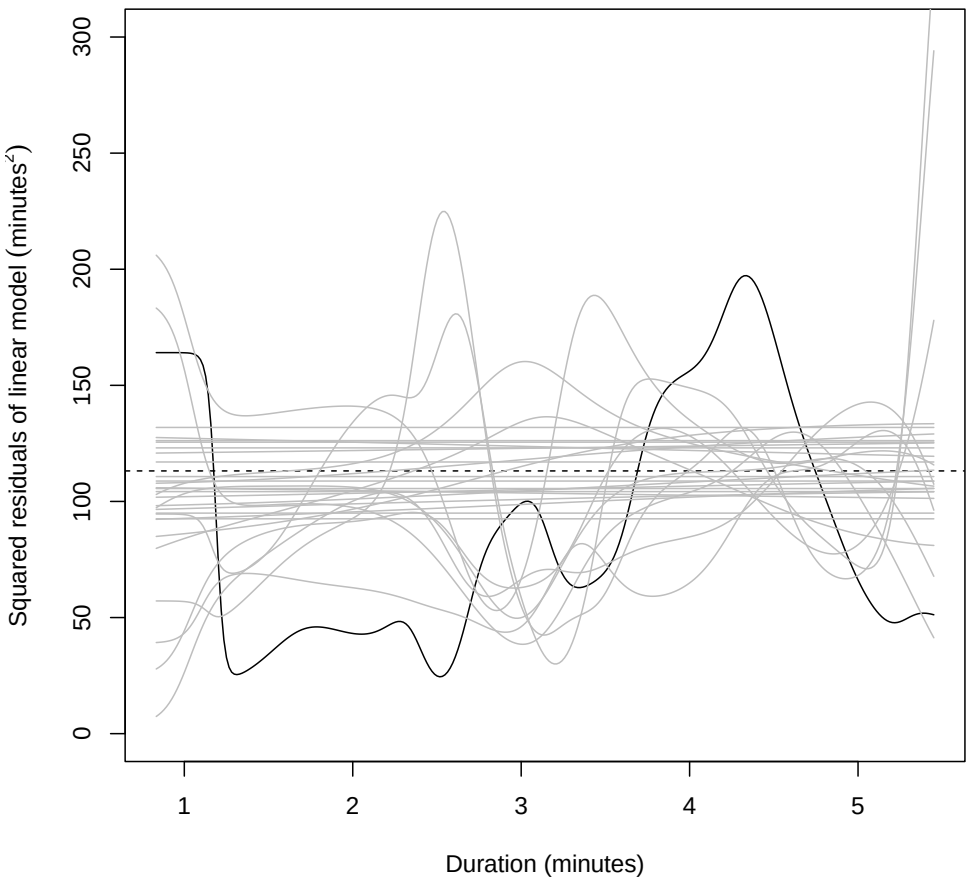
While that sort of qualitative comparison is genuinely informative, one can also be more quantitative. One might measure heteroskedasticity by, say, evaluating the conditional variance at all the data points, and looking at the ratio of the interquartile range to the median. This would be zero for perfect homoskedasticity, and grow as the dispersion of actual variances around the "typical" variance increased. For the data, this is `IQR(fitted(geyser.var))/median(fitted(geyser.var))` =. Simulations from the OLS model give values around 10^{-15}.

There is nothing particularly special about this measure of heteroskedasticity — after all, I just made it up. The broad point it illustrates is the one made in §5.4.2.1: whenever we have some sort of quantitative summary statistic we can calculate on our real data, we can also calculate the same statistic on realizations of the model, and the difference will then tell us something about how close the simulations, and so the model, come to the data. In this case, we learn that the linear, homoskedastic model seriously understates the variability of this data. That leaves open the question of whether the problem is the linearity or the homoskedasticity; I will leave that question to Exercise 10.6.




```
library(MASS)
data(geyser)
geyser.ols <- lm(waiting ~ duration, data = geyser)
plot(geyser$duration, residuals(geyser.ols)^2, cex = 0.5, pch = 16, xlab = "Duration (minutes)",
    ylab = expression(`Squared residuals of linear model `(minutes^2)))
geyser.var <- npreg(residuals(geyser.ols)^2 ~ geyser$duration)
duration.order <- order(geyser$duration)
lines(geyser$duration[duration.order], fitted(geyser.var)[duration.order])
abline(h = summary(geyser.ols)$sigma^2, lty = "dashed")
legend("topleft", legend = c("data", "kernel variance", "homoskedastic (OLS)"), lty = c("blank",
    "solid", "dashed"), pch = c(16, NA, NA))
```

Figure 10.9 Squared residuals from the linear model of Figure 5.1, plotted against duration, along with the unconditional, homoskedastic variance implicit in OLS (dashed), and a kernel-regression estimate of the conditional variance (solid).




```
duration.grid <- seq(from = min(geyser$duration), to = max(geyser$duration), length.out = 300)
plot(duration.grid, predict(geyser.var, exdat = duration.grid), ylim = c(0, 300),
    type = "l", xlab = "Duration (minutes)", ylab = expression(`Squared residuals of linear model `(
abline(h = summary(geyser.ols)$sigma^2, lty = "dashed")
one.var.func <- function() {
    fit <- lm(waiting ~ duration, data = rgeyser())
    var.func <- npreg(residuals(fit)^2 ~ geyser$duration)
    lines(duration.grid, predict(var.func, exdat = duration.grid), col = "grey")
}
invisible(replicate(30, one.var.func()))
```

Figure 10.10 The actual conditional variance function estimated from the Old Faithful data (and the linear regression), in black, plus the results of applying the same procedure to simulations from the homoskedastic linear regression model (grey lines; see §5.4.2 for the `rgeyser` function). The fact that the estimates from the simulations are mostly flat or gently sloped suggests that the changes in variance found in the data are likely too large to just be sampling noise.

10.4 Re-sampling Residuals with Heteroskedasticity

Re-sampling the residuals of a regression, as described in §6.4, assumes that the distribution of fluctuations around the regression curve is the same for all values of the input x. Under heteroskedasticity, this is of course not the case. Nonetheless, we can still re-sample residuals to get bootstrap confidence intervals, standard errors, and so forth, provided we define and scale them properly. If we have a conditional variance function $\hat{\sigma}^2(x)$, as well as the estimated regression function $\widehat{\mu}(x)$, we can combine them to re-sample heteroskedastic residuals.

1. Construct the standardized residuals, by dividing the actual residuals by the conditional standard deviation:
$$\eta_i = \epsilon_i / \hat{\sigma}(x_i) \tag{10.29}$$
The η_i should now be all the same magnitude (in distribution!), no matter where x_i is in the space of predictors.
2. Re-sample the η_i with replacement, to get $\tilde{\eta}_1, \ldots \tilde{\eta}_n$.
3. Set $\tilde{x}_i = x_i$.
4. Set $\tilde{y}_i = \widehat{\mu}(\tilde{x}_i) + \hat{\sigma}(\tilde{x}_i)\tilde{\eta}_i$.
5. Analyze the surrogate data $(\tilde{x}_1, \tilde{y}_1), \ldots (\tilde{x}_n, \tilde{y}_n)$ like it was real data.

Of course, this still assumes that the *only* difference in distribution for the noise at different values of x is the scale.

10.5 Local Linear Regression

Switching gears, recall from Chapter 2 that one reason it can be sensible to use a linear approximation to the true regression function μ is that we can typically Taylor-expand (App. B) the latter around any point x_0,

$$\mu(x) = \mu(x_0) + \sum_{k=1}^{\infty} \frac{(x-x_0)^k}{k!} \left. \frac{d^k \mu}{dx^k} \right|_{x=x_0} \tag{10.30}$$

and similarly with all the partial derivatives in higher dimensions. Truncating the series at first order, $\mu(x) \approx \mu(x_0) + (x-x_0)\mu'(x_0)$, we see the first derivative $\mu'(x_0)$ is the best linear prediction coefficient, at least if x close enough to x_0. The snag in this line of argument is that if $\mu(x)$ is nonlinear, then μ' isn't a constant, and the optimal linear predictor changes depending on where we want to make predictions.

However, statisticians are thrifty people, and having assembled all the machinery for linear regression, they are loathe to throw it away just because the fundamental model is wrong. If we can't fit one line, why not fit many? If each point has a different best linear regression, why not estimate them all? Thus the idea of **local** linear regression: fit a different linear regression everywhere, weighting the data points by how close they are to the point of interest[6].

The simplest approach we could take would be to divide up the range of x into so many bins, and fit a separate linear regression for each bin. This has at least three drawbacks. First, we get weird discontinuities at the boundaries between bins. Second, we induce an odd sort of bias, where our predictions near the boundaries of a bin depend strongly on data from one side of the bin, and not at all on nearby data points just across the border. Third, we need to pick the bins.

The next simplest approach would be to first figure out where we want to make a prediction (say x), and do a linear regression with all the data points which were sufficiently close, $|x_i - x| \leq h$ for some h. Now we are basically using a uniform-density kernel to weight the data points. This eliminates two problems from the binning idea — the examples we include are always centered on the x we're trying to get a prediction for, and we just need to pick one bandwidth h rather than placing all the bin boundaries. But still, each example point always has either weight 0 or weight 1, so our predictions change jerkily as training points fall into or out of the window. It generally works nicer to have the weights change more smoothly with the distance, starting off large and then gradually trailing to zero.

By now bells may be going off, as this sounds very similar to the kernel regression. In fact, kernel regression is what happens when we truncate Eq. 10.30 at *zeroth* order, getting **locally constant** regression. We set up the problem

$$\widehat{\mu}(x) = \operatorname*{argmin}_m \frac{1}{n} \sum_{i=1}^{n} w_i(x)(y_i - m)^2 \tag{10.31}$$

[6] Some people say "local linear" and some "local*ly* linear".

and get the solution

$$\widehat{\mu}(x) = \sum_{i=1}^{n} y_i \frac{w_i(x)}{\sum_{j=1}^{n} w_j(x)} \tag{10.32}$$

which just is our kernel regression, when the weights are proportional to the kernels, $w_i(x) \propto K(x_i, x)$. (Without loss of generality, we can take the constant of proportionality to be 1.)

What about **locally linear** regression? The optimization problem is

$$\left(\widehat{\mu}(x), \widehat{\beta}(x)\right) = \underset{m,\beta}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} w_i(x)(y_i - m - (x_i - x) \cdot \beta)^2 \tag{10.33}$$

where again we can make $w_i(x)$ proportional to some kernel function, $w_i(x) \propto K(x_i, x)$. To solve this, abuse notation slightly to define $z_i = (1, x_i - x)$, i.e., the displacement from x, with a 1 stuck at the beginning to (as usual) handle the intercept. Now, by the machinery above,

$$\left(\widehat{\mu}(x), \widehat{\beta}(x)\right) = (\mathbf{z}^T \mathbf{w}(x) \mathbf{z})^{-1} \mathbf{z}^T \mathbf{w}(x) \mathbf{y} \tag{10.34}$$

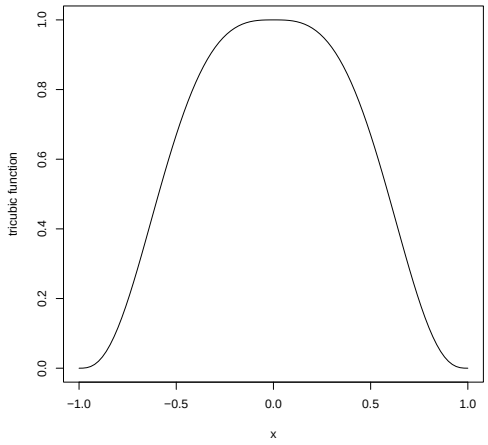
and the prediction is just the intercept, $\widehat{\mu}(x)$. If you need an estimate of the first derivatives, those are the $\widehat{\beta}(x)$. Eq. 10.34 guarantees that the weights given to each training point change smoothly with x, so the predictions will also change smoothly.[7]

Using a smooth kernel whose density is positive everywhere, like the Gaussian, ensures that the weights will change smoothly. But we could also use a kernel which goes to zero outside some finite range, so long as the kernel rises gradually from zero inside the range. For locally linear regression, a common choice of kernel is therefore the **tri-cubic**,

$$K(x_i, x) = \left(1 - \left(\frac{|x_i - x_0|}{h}\right)^3\right)^3 \tag{10.35}$$

if $|x - x_i| < h$, and $= 0$ otherwise (Figure 10.11).

[7] Notice that local linear predictors are still linear smoothers as defined in Chapter 1, (i.e., the predictions are linear in the y_i), but they are not, strictly speaking, *kernel* smoothers, since you can't re-write the last equation in the form of a kernel average.




```
curve((1 - abs(x)^3)^3, from = -1, to = 1, ylab = "tricubic function")
```

Figure 10.11 The tricubic kernel, with broad plateau where $|x| \approx 0$, and the smooth fall-off to zero at $|x| = 1$.

10.5.1 For and Against Locally Linear Regression

Why would we use locally linear regression, if we already have kernel regression?

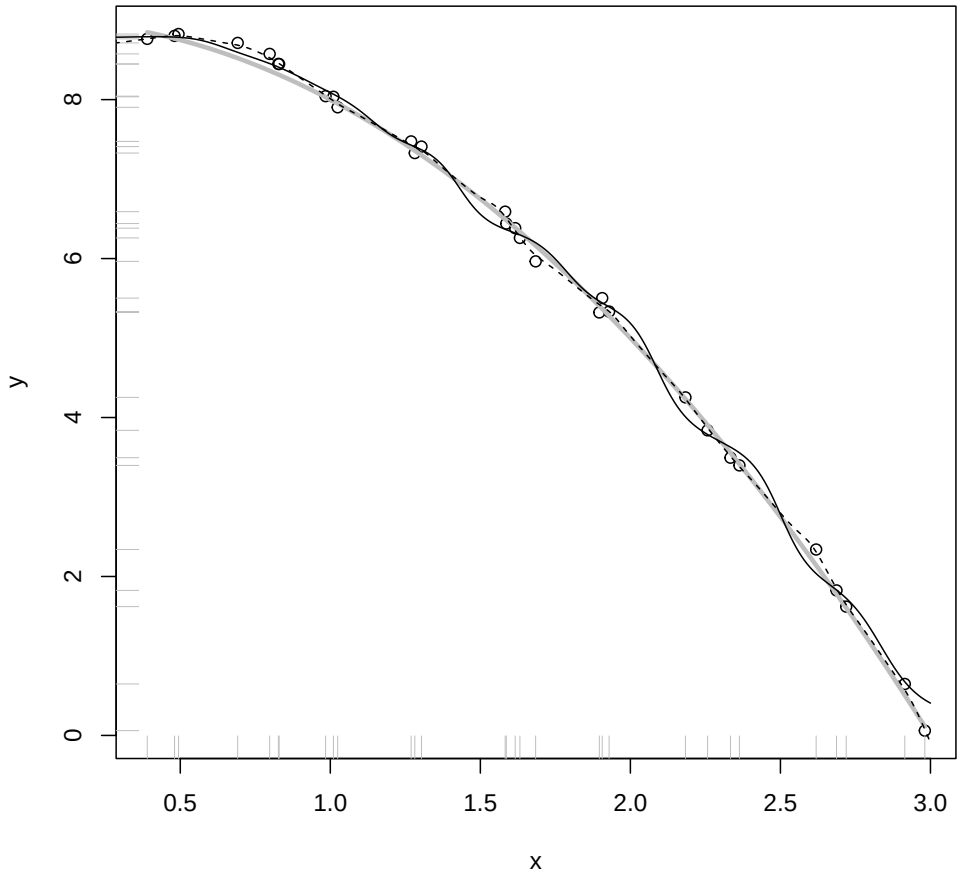
1. You may recall that when we worked out the bias of kernel smoothers (Eq. 4.10 in Chapter 4), we got a contribution that was proportional to $\mu'(x)$. If we do an analogous analysis for locally linear regression, the bias is the same, *except* that this derivative term goes away.
2. Relatedly, that analysis we did of kernel regression tacitly assumed the point we were looking at was in the middle of the training data (or at least rather more than h from the border). The bias gets worse near the edges of the training data. Suppose that the true $\mu(x)$ is decreasing in the vicinity of the largest x_i. (See the grey curve in Figure 10.12.) When we make our predictions there, in kernel regression we can only average values of y_i which tend to be systematically larger than the value we want to predict. This means that our kernel predictions are systematically biased upwards, and the size of the bias grows with $\mu'(x)$. (See the black line in Figure 10.12 at the lower right.) If we use a locally linear model, however, it can pick up that there is a trend, and reduce the edge bias by extrapolating it (dashed line in the figure).
3. The predictions of locally linear regression tend to be smoother than those of kernel regression, simply because we are locally fitting a smooth line rather than a flat constant. As a consequence, estimates of the derivative $\frac{d\widehat{\mu}}{dx}$ tend to be less noisy when $\widehat{\mu}$ comes from a locally linear model than a kernel regression.

Of course, total prediction error depends not only on the bias but also on the variance. Remarkably enough, the variance for kernel regression and locally linear regression is the same, at least asymptotically. Since locally linear regression has smaller bias, local-linear fits are often better predictors.

Despite all these advantages, local linear models have a real drawback. To make a prediction with a kernel smoother, we have to calculate a weighted average. To make a prediction with a local linear model, we have to solve a (weighted) linear least squares problem for each point, or each prediction. This takes much more computing time[8].

There are several packages which implement locally linear regression. Since we are already using `np`, one of the simplest is to set the `regtype="ll"` in

[8] Let's think this through. To find $\widehat{\mu}(x)$ with a kernel smoother, we need to calculate $K(x_i, x)$ for each x_i. If we've got p predictor variables and use a product kernel, that takes $O(pn)$ computational steps. We then need to add up the kernels to get the denominator, which we could certainly do in $O(n)$ more steps. (Could you do it faster?) Multiplying each weight by its y_i is a further $O(n)$, and the final adding up is at most $O(n)$; total, $O(pn)$. To make a prediction with a local linear model, we need to calculate the right-hand side of Eq. 10.34. Finding $(\mathbf{z}^T\mathbf{w}(x)\mathbf{z})$ means multiplying $[(p+1)\times n][n\times n][n\times(p+1)]$ matrices, which will take $O((p+1)^2 n) = O(p^2 n)$ steps. Inverting a $q\times q$ matrix takes $O(q^3)$ steps, so our inversion takes $O((p+1)^3) = O(p^3)$ steps. Just getting $(\mathbf{z}^T\mathbf{w}(x)\mathbf{z})^{-1}$ thus requires $O(p^3 + p^2 n)$. Finding the $(p+1)\times 1$ matrix $\mathbf{z}^T\mathbf{w}(x)\mathbf{y}$ similarly takes $O((p+1)n) = O(pn)$ steps, and the final matrix multiplication is $O((p+1)(p+1)) = O(p^2)$. Total, $O(p^2 n) + O(p^3)$. The speed advantage of kernel smoothing thus gets increasingly extreme as the number of predictor variables p grows.




```
x <- runif(30, max = 3)
y <- 9 - x^2 + rnorm(30, sd = 0.1)
plot(x, y)
rug(x, side = 1, col = "grey")
rug(y, side = 2, col = "grey")
curve(9 - x^2, col = "grey", add = TRUE, lwd = 3)
grid.x <- seq(from = 0, to = 3, length.out = 300)
np0 <- npreg(y ~ x)
lines(grid.x, predict(np0, exdat = grid.x))
np1 <- npreg(y ~ x, regtype = "ll")
lines(grid.x, predict(np1, exdat = grid.x), lty = "dashed")
```

Figure 10.12 Points are samples from the true, nonlinear regression function shown in grey. The solid black line is a kernel regression, and the dashed line is a locally linear regression. Note that the locally linear model is smoother than the kernel regression, and less biased when the true curve has a non-zero bias at a boundary of the data (far right).

`npreg`.[9] There are several other packages which support it, notably `KernSmooth` and `locpoly`.

As the name of the latter suggests, there is no reason we *have* to stop at locally linear models, and we could use local polynomials of any order. The main reason to use a higher-order local polynomial, rather than a locally-linear or locally-constant model, is to estimate higher derivatives. Since this is a somewhat specialized topic, I will not say more about it.

[9] `"ll"` stands for "locally linear", of course; the default is `regtype="lc"`, for "locally constant".

10.5.2 Lowess

There is however one additional topic in locally linear models which is worth mentioning. This is the variant called **lowess** or **loess**.[10] The basic idea is to fit a locally linear model, with a kernel which goes to zero outside a finite window and rises gradually inside it, typically the tri-cubic I plotted earlier. The wrinkle, however, is that rather than solving a least *squares* problem, it minimizes a different and more "robust" loss function,

$$\underset{\beta(x)}{\operatorname{argmin}} \frac{1}{n}\sum_{i=1}^{n} w_i(x)\ell(y - \vec{x}_i \cdot \beta(x)) \tag{10.36}$$

where $\ell(a)$ doesn't grow as rapidly for large a as a^2. The idea is to make the fitting less vulnerable to occasional large outliers, which would have very large squared errors, unless the regression curve went far out of its way to accommodate them. For instance, we might have $\ell(a) = a^2$ if $|a| < 1$, and $\ell(a) = 2|a| - 1$ otherwise[11]. There is a large theory of robust estimation, largely parallel to the more familiar least-squares theory. In the interest of space, we won't pursue it further, but lowess is worth mentioning because it's such a common smoothing technique, especially for sheer visualization.

Lowess smoothing is implemented in base R through the function `lowess` (rather basic), and through the function `loess` (more sophisticated), as well as in the CRAN package `locfit` (more sophisticated still). The lowess idea can be combined with local fitting of higher-order polynomials; the `loess` and `locfit` commands both support this.

[10] I have heard this name explained as an acronym for both "locally weighted scatterplot smoothing" and "locally weight sum of squares".

[11] This is called the **Huber loss**; it continuously interpolates between looking like squared error and looking like absolute error. This means that when errors are small, it gives results very like least-squares, but it is resistant to outliers. See also App. J.6.1.

10.6 Further Reading

Weighted least squares goes back to the 19th century, almost as far back as ordinary least squares; see the references in chapter 1 and 2.

I am not sure who invented the use of smoothing to estimate variance functions; I learned it from Wasserman (2006, pp. 87–88). I've occasionally seen it done with a linear model for the conditional variance, which I can't recommend.

Simonoff (1996) is a good reference on local linear and local polynomial models, including actually doing the bias-variance analyses where I've just made empty "it can be shown" promises. Fan and Gijbels (1996) is more comprehensive, but also a much harder read. Lowess was introduced by Cleveland (1979), but the name evidently came later (since it doesn't appear in that paper).

Exercises

10.1 Imagine we are trying to estimate the mean value of Y from a large population of size n_0, so $\overline{y} = n_0^{-1} \sum_{j=1}^{n} y_j$. We observe $n \ll n_0$ members of the population, with individual i being included in our sample with a probability proportional to π_i.

1. Show that $\left(\sum_{i=1}^{n} y_i/\pi_i\right) / \sum_{i'=1}^{n} 1/\pi_{i'}$ is a consistent estimator of $\overline{y}$, by showing that that it is unbiased and it has a variance that shrinks with n towards 0.
2. Is the unweighted sample mean $n^{-1} \sum_{i=1}^{n} y_i$ a consistent estimator of $\overline{y}$ when the π_i are not all equal?

10.2 Show that the model of Eq. 10.13 has the log-likelihood given by Eq. 10.14

10.3 Do the calculus to verify Eq. 10.4.

10.4 Is $w_i = 1$ a necessary as well as a sufficient condition for Eq. 10.3 and Eq. 10.1 to have the same minimum?

10.5 §10.2.2 showed that WLS gives better parameter estimates than OLS when there is heteroskedasticity, and we know and use the variance. Modify the code for to see which one has better generalization error.

10.6 §10.3.2 looked at the residuals of the linear regression model for the Old Faithful geyser data, and showed that they would imply lots of heteroskedasticity. This might, however, be an artifact of inappropriately using a linear model. Use either kernel regression (cf. §6.4.2) or local linear regression to estimate the conditional mean of waiting given duration, and see whether the apparent heteroskedasticity goes away.

10.7 Should local linear regression do better or worse than ordinary least squares under heteroskedasticity? What exactly would this mean, and how might you test your ideas?

11

Logistic Regression

11.1 Modeling Conditional Probabilities

So far, we either looked at estimating the conditional expectations of continuous variables (as in regression), or at estimating distributions. There are many situations where however we are interested in input-output relationships, as in regression, but the output variable is discrete rather than continuous. In particular there are many situations where we have binary outcomes (it snows in Pittsburgh on a given day, or it doesn't; this squirrel carries plague, or it doesn't; this loan will be paid back, or it won't; this person will get heart disease in the next five years, or they won't). In addition to the binary outcome, we have some input variables, which may or may not be continuous. How could we model and analyze such data?

We could try to come up with a rule which guesses the binary output from the input variables. This is called **classification**, and is an important topic in statistics and machine learning. However, guessing "yes" or "no" is pretty crude — especially if there is no perfect rule. (Why should there be a perfect rule?) Something which takes noise into account, and doesn't just give a binary answer, will often be useful. In short, we want probabilities — which means we need to fit a stochastic model.

What would be nice, in fact, would be to have conditional distribution of the response Y, given the input variables, $\Pr(Y|X)$. This would tell us about how precise our predictions should be. If our model says that there's a 51% chance of snow and it doesn't snow, that's better than if it had said there was a 99% chance of snow (though even a 99% chance is not a sure thing). We will see, in Chapter 14, general approaches to estimating conditional probabilities non-parametrically, which can use the kernels for discrete variables from Chapter 4. While there are a lot of merits to this approach, it does involve coming up with a model for the joint distribution of outputs Y and inputs X, which can be quite time-consuming.

Let's pick one of the classes and call it "1" and the other "0". (It doesn't matter which is which.) Then Y becomes an **indicator variable**, and you can convince yourself that $\Pr(Y = 1) = \mathbb{E}[Y]$. Similarly, $\Pr(Y = 1|X = x) = \mathbb{E}[Y|X = x]$. (In a phrase, "conditional probability is the conditional expectation of the indicator".) This helps us because by this point we know all about estimating conditional expectations. The most straightforward thing for us to do at this point

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

would be to pick out our favorite smoother and estimate the regression function for the indicator variable; this will be an estimate of the conditional probability function.

There are two reasons not to just plunge ahead with that idea. One is that probabilities must be between 0 and 1, but our smoothers will not necessarily respect that, even if all the observed y_i they get are either 0 or 1. The other is that we might be better off making more use of the fact that we are trying to estimate probabilities, by more explicitly modeling the probability.

Assume that $\Pr(Y = 1|X = x) = p(x;\theta)$, for some function p parameterized by θ. parameterized function θ, *and* further assume that observations are independent of each other. The the (conditional) likelihood function is

$$\prod_{i=1}^{n} \Pr(Y = y_i|X = x_i) = \prod_{i=1}^{n} p(x_i;\theta)^{y_i}(1 - p(x_i;\theta))^{1-y_i} \tag{11.1}$$

Recall that in a sequence of Bernoulli trials $y_1, \ldots y_n$, where there is a constant probability of success p, the likelihood is

$$\prod_{i=1}^{n} p^{y_i}(1-p)^{1-y_i} \tag{11.2}$$

As you learned in basic statistics, this likelihood is maximized when $p = \hat{p} = n^{-1}\sum_{i=1}^{n} y_i$. If each trial had its own success probability p_i, this likelihood becomes

$$\prod_{i=1}^{n} p_i^{y_i}(1-p_i)^{1-y_i} \tag{11.3}$$

Without some constraints, estimating the "inhomogeneous Bernoulli" model by maximum likelihood doesn't work; we'd get $\hat{p}_i = 1$ when $y_i = 1$, $\hat{p}_i = 0$ when $y_i = 0$, and learn nothing. If on the other hand we assume that the p_i aren't just arbitrary numbers but are linked together, if we *model* the probabilities, those constraints give non-trivial parameter estimates, and let us generalize. In the kind of model we are talking about, the constraint, $p_i = p(x_i;\theta)$, tells us that p_i must be the same whenever x_i is the same, and if p is a continuous function, then similar values of x_i must lead to similar values of p_i. Assuming p is known (up to parameters), the likelihood is a function of θ, and we can estimate θ by maximizing the likelihood. This chapter will be about this approach.

11.2 Logistic Regression

To sum up: we have a binary output variable Y, and we want to model the conditional probability $\Pr(Y = 1|X = x)$ as a function of x; any unknown parameters in the function are to be estimated by maximum likelihood. By now, it will not surprise you to learn that statisticians have approached this problem by asking themselves "how can we use linear regression to solve this?"

1. The most obvious idea is to let $p(x)$ be a linear function of x. Every increment of a component of x would add or subtract so much to the probability. This is called a "linear probability model". The conceptual problem here is that p must be between 0 and 1, and linear functions are unbounded. Moreover, in many situations we empirically see "diminishing returns" — changing p by the same amount requires a bigger change in x when p is already large (or small) than when p is close to 1/2. Linear models can't do this.
2. The next most obvious idea is to let $\log p(x)$ be a linear function of x, so that changing an input variable *multiplies* the probability by a fixed amount. The problem is that logarithms of probabilities are unbounded in only one direction, and linear functions are not.
3. Finally, the easiest modification of $\log p$ which has an unbounded range is the **logistic transformation** (or **logit**) , $\log \frac{p}{1-p}$. We can make *this* a linear function of x without fear of nonsensical results. (Of course the results could still happen to be *wrong*, but they're not *guaranteed* to be wrong.)

This last alternative is **logistic regression**.

Formally, the logistic regression model is that

$$\log \frac{p(x)}{1-p(x)} = \beta_0 + x \cdot \beta \tag{11.4}$$

Solving for p, this gives

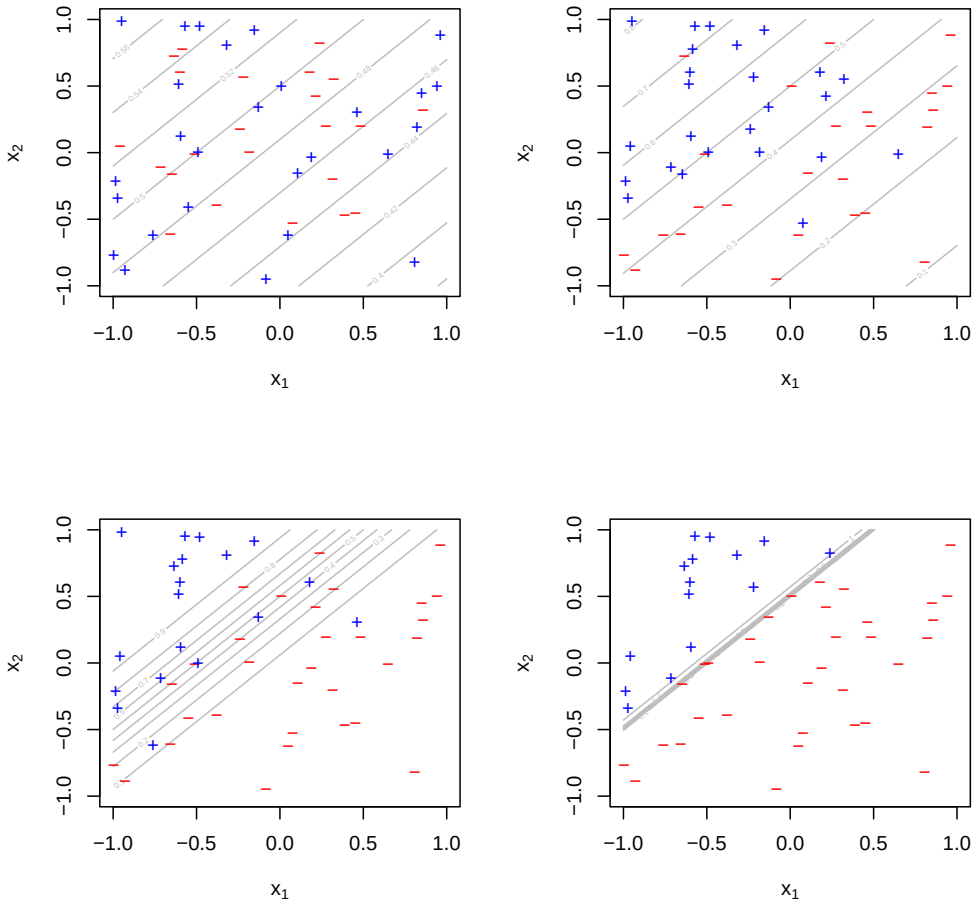
$$p(x;\beta) = \frac{e^{\beta_0 + x\cdot\beta}}{1+e^{\beta_0+x\cdot\beta}} = \frac{1}{1+e^{-(\beta_0+x\cdot\beta)}} \tag{11.5}$$

Notice that the overall specification is a lot easier to grasp in terms of the transformed probability that in terms of the untransformed probability.[1]

To minimize the mis-classification rate, we should predict $Y = 1$ when $p \geq 0.5$ and $Y = 0$ when $p < 0.5$ (Exercise 11.1). This means guessing 1 whenever $\beta_0 + x\cdot\beta$ is non-negative, and 0 otherwise. So logistic regression gives us a **linear classifier**. The **decision boundary** separating the two predicted classes is the solution of $\beta_0 + x\cdot\beta = 0$, which is a point if x is one dimensional, a line if it is two dimensional, etc. One can show (exercise!) that the distance from the decision boundary is $\beta_0/\|\beta\| + x\cdot\beta/\|\beta\|$. Logistic regression not only says where the boundary between the classes is, but also says (via Eq. 11.5) that the class probabilities depend on distance from the boundary, in a particular way, and that they go towards the extremes (0 and 1) more rapidly when $\|\beta\|$ is larger. It's these statements about probabilities which make logistic regression more than just a classifier. It makes stronger, more detailed predictions, and can be fit in a different way; but those strong predictions could be wrong.

Using logistic regression to predict class probabilities is a *modeling choice*, just like it's a modeling choice to predict quantitative variables with linear regression. In neither case is the appropriateness of the model guaranteed by the gods, nature,

[1] Unless you've taken thermodynamics or physical chemistry, in which case you recognize that this is the Boltzmann distribution for a system with two states, which differ in energy by $\beta_0 + x \cdot \beta$.




```
x <- matrix(runif(n = 50 * 2, min = -1, max = 1), ncol = 2)
par(mfrow = c(2, 2))
plot.logistic.sim(x, beta.0 = -0.1, beta = c(-0.2, 0.2))
y.1 <- plot.logistic.sim(x, beta.0 = -0.5, beta = c(-1, 1))
plot.logistic.sim(x, beta.0 = -2.5, beta = c(-5, 5))
plot.logistic.sim(x, beta.0 = -250, beta = c(-500, 500))
```

Figure 11.1 Effects of scaling logistic regression parameters. Values of x_1 and x_2 are the same in all plots ($\sim \text{Unif}(-1, 1)$ for both coordinates), but labels were generated randomly from logistic regressions with $\beta_0 = -0.1$, $\beta = (-0.2, 0.2)$ (top left); from $\beta_0 = -0.5$, $\beta = (-1, 1)$ (top right); from $\beta_0 = -2.5$, $\beta = (-5, 5)$ (bottom left); and from $\beta_0 = 2.5 \times 10^2$, $\beta = (-5 \times 10^2, 5 \times 10^2)$. Notice how as the parameters get increased in constant ratio to each other, we approach a deterministic relation between Y and x, with a linear boundary between the classes. (We save one set of the random binary responses for use later, as the imaginatively-named `y.1`.)

```
sim.logistic <- function(x, beta.0, beta, bind = FALSE) {
    require(faraway)
    linear.parts <- beta.0 + (x %*% beta)
    y <- rbinom(nrow(x), size = 1, prob = ilogit(linear.parts))
    if (bind) {
        return(cbind(x, y))
    }
    else {
        return(y)
    }
}

plot.logistic.sim <- function(x, beta.0, beta, n.grid = 50, labcex = 0.3, col = "grey",
    ...) {
    grid.seq <- seq(from = -1, to = 1, length.out = n.grid)
    plot.grid <- as.matrix(expand.grid(grid.seq, grid.seq))
    require(faraway)
    p <- matrix(ilogit(beta.0 + (plot.grid %*% beta)), nrow = n.grid)
    contour(x = grid.seq, y = grid.seq, z = p, xlab = expression(x[1]), ylab = expression(x[2]),
        main = "", labcex = labcex, col = col)
    y <- sim.logistic(x, beta.0, beta, bind = FALSE)
    points(x[, 1], x[, 2], pch = ifelse(y == 1, "+", "-"), col = ifelse(y == 1, "blue",
        "red"))
    invisible(y)
}
```

Code Example 26: *Code to simulate binary responses from a logistic regression model, and to plot a 2D logistic regression's probability contours and simulated binary values. (How would you modify this to take the responses from a data frame?*

mathematical necessity, etc. We begin by positing the model, to get something to work with, and we end (if we know what we're doing) by checking whether it really does match the data, or whether it has systematic flaws.

Logistic regression is one of the most commonly used tools for applied statistics and discrete data analysis. There are basically four reasons for this.

1. Tradition.
2. In addition to the heuristic approach above, the quantity $\log p/(1-p)$ plays an important role in the analysis of contingency tables (the "log odds"). Classification is a bit like having a contingency table with two columns (classes) and infinitely many rows (values of x). With a finite contingency table, we can estimate the log-odds for each row empirically, by just taking counts in the table. With infinitely many rows, we need some sort of interpolation scheme; logistic regression is linear interpolation for the log-odds.
3. It's closely related to "exponential family" distributions, where the probability of some vector v is proportional to $\exp\left\{\beta_0 + \sum_{j=1}^{m} f_j(v)\beta_j\right\}$. If one of the components of v is binary, and the functions f_j are all the identity function, then we get a logistic regression. Exponential families arise in many contexts in statistical theory (and in physics!), so there are lots of problems which can be turned into logistic regression.

4. It often works surprisingly well as a classifier. But, many simple techniques often work surprisingly well as classifiers, and this doesn't really testify to logistic regression getting the probabilities right.

11.2.1 Likelihood Function for Logistic Regression

[[TODO: Standardize notation here for likelihood function compared to theory appendix]]

Because logistic regression predicts probabilities, rather than just classes, we can fit it using likelihood. For each training data-point, we have a vector of features, x_i, and an observed class, y_i. The probability of that class was either p, if $y_i = 1$, or $1 - p$, if $y_i = 0$. The likelihood is then

$$L(\beta_0, \beta) = \prod_{i=1}^{n} p(x_i)^{y_i}(1 - p(x_i))^{1-y_i} \tag{11.6}$$

(I could substitute in the actual equation for p, but things will be clearer in a moment if I don't.) The log-likelihood turns products into sums:

$$\ell(\beta_0, \beta) = \sum_{i=1}^{n} y_i \log p(x_i) + (1 - y_i) \log (1 - p(x_i)) \tag{11.7}$$

$$= \sum_{i=1}^{n} \log (1 - p(x_i)) + \sum_{i=1}^{n} y_i \log \frac{p(x_i)}{1 - p(x_i)} \tag{11.8}$$

$$= \sum_{i=1}^{n} \log (1 - p(x_i)) + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta) \tag{11.9}$$

$$= \sum_{i=1}^{n} - \log \left(1 + e^{\beta_0 + x_i \cdot \beta}\right) + \sum_{i=1}^{n} y_i(\beta_0 + x_i \cdot \beta) \tag{11.10}$$

where in the next-to-last step we finally use equation 11.4.

Typically, to find the maximum likelihood estimates we'd differentiate the log likelihood with respect to the parameters, set the derivatives equal to zero, and solve. To start that, take the derivative with respect to one component of β, say β_j.

$$\frac{\partial \ell}{\partial \beta_j} = -\sum_{i=1}^{n} \frac{1}{1 + e^{\beta_0 + x_i \cdot \beta}} e^{\beta_0 + x_i \cdot \beta} x_{ij} + \sum_{i=1}^{n} y_i x_{ij} \tag{11.11}$$

$$= \sum_{i=1}^{n} \left(y_i - p(x_i; \beta_0, \beta)\right) x_{ij} \tag{11.12}$$

We are not going to be able to set this to zero and solve exactly. (That's a transcendental equation, and there is no closed-form solution.) We can however approximately solve it numerically.

11.3 Numerical Optimization of the Likelihood

While our likelihood isn't nice enough that we have an explicit expression for the maximum (the way we do in OLS or WLS), it is a pretty well-behaved function, and one which is amenable to lots of the usual numerical methods for optimization. In particular, like most log-likelihood functions, it's suitable for an application of Newton's method. Briefly (see Appendix D.2 for details), Newton's method starts with an initial guess about the optimal parameters, and then calculates the gradient of the log-likelihood with respect to those parameters. It then adds an amount proportional to the gradient to the parameters, moving up the surface of the log-likelihood function. The size of the step in the gradient direction is dictated by the second derivatives — it takes bigger steps when the second derivatives are small (so the gradient is a good guide to what the function looks like), and small steps when the curvature is large.

11.3.1 Iteratively Re-Weighted Least Squares

Remarkably enough, in the case of logistic regression, each step of Newton's method ends up looking like a good, old-fashioned linear regression problem.

Fundamentally, this is because logistic regression is a linear model for a transformation of the probability. Let's call this transformation g:

$$g(p) \equiv \log \frac{p}{1-p} \tag{11.13}$$

So the model is

$$g(p) = \beta_0 + x \cdot \beta \tag{11.14}$$

and $Y|X = x \sim \text{Binom}(1, g^{-1}(\beta_0 + x \cdot \beta))$. It seems that what we should want to do is take $g(y)$ and regress it linearly on x. Of course, the variance of Y, according to the model, is going to change depending on x — it will be $(g^{-1}(\beta_0 + x \cdot \beta))(1 - g^{-1}(\beta_0 + x \cdot \beta))$ — so we really ought to do a weighted linear regression, with weights inversely proportional to that variance. (We learned about weighted linear regression in Chapter 10.) Since writing $g^{-1}(\beta_0 + x \cdot \beta)$ is getting annoying, let's abbreviate it by $p(x)$ or just p, and let's abbreviate that variance as $V(p)$.

The problem is that y is either 0 or 1, so $g(y)$ is either $-\infty$ or $+\infty$. We will evade this by using first-order Taylor expansion (App. B).

$$g(y) \approx g(p) + (y - p)g'(p) \equiv z \tag{11.15}$$

The right hand side, z will be our *effective* response variable, which we will regress on x. To see why this should give us the right coefficients, substitute for $g(p)$ in the definition of z,

$$z = \beta_0 + x \cdot \beta + (y - p)g'(p) \tag{11.16}$$

and notice that, if we've got the coefficients right, $\mathbb{E}[Y|X = x] = p$, so $(y - p)$ should be mean-zero noise. In other words, when we have the right coefficients, z *is* a linear function of x plus mean-zero noise. (This is our excuse for throwing

away the rest of the Taylor expansion, even though we know the discarded terms are infinitely large!) That noise doesn't have constant variance, but we can work it out,

$$\mathbb{V}[Z|X=x] = \mathbb{V}[(Y-p)g'(p)|X=x] = (g'(p))^2 V(p) \ , \tag{11.17}$$

and so use that variance in weighted least squares to recover β.

Notice that z and the weights both involve the parameters of our logistic regression, through $p(x)$. So having done this once, we should really use the new parameters to update z and the weights, and do it again. Eventually, we come to a fixed point, where the parameter estimates no longer change. This loop — start with a guess about the parameters, use it to calculate the z_i and their weights, regress on the x_i to get new parameters, and repeat — is known as **iterative reweighted least squares** (IRLS or IRWLS), **iterative weighted least squares** (IWLS), etc.

The treatment above is rather heuristic[2], but it turns out to be equivalent to using Newton's method, only with the expected second derivative of the log likelihood, instead of its actual value. This takes a reasonable amount of algebra to show, so we'll skip it (but see Exercise 11.3)[3]. Since, with a large number of observations, the observed second derivative should be close to the expected second derivative, this is only a small approximation.

Perfect Classification

One caution about using maximum likelihood to fit logistic regression is that it can seem to work badly when the training data *can* be linearly separated. The reason is that, to make the likelihood large, $p(x_i)$ should be large when $y_i = 1$, and p should be small when $y_i = 0$. If β_0, β_0 is a set of parameters which perfectly classifies the training data, then $c\beta_0, c\beta$ is too, for any $c > 1$, but in a logistic regression the second set of parameters will have more extreme probabilities, and so a higher likelihood. For linearly separable data, then, there is no parameter vector which *maximizes* the likelihood, since ℓ can always be increased by making the vector larger but keeping it pointed in the same direction.

You should, of course, be so lucky as to have this problem.

[2] That is, mathematically incorrect.

[3] The two key points are as follows. First, the gradient of the log-likelihood turns out to be the sum of the $z_i x_i$. (Cf. Eq. 11.12.) Second, take a single Bernoulli observation with success probability p. The log-likelihood is $Y \log p + (1-Y)\log 1 - p$. The first derivative with respect to p is $Y/p - (1-Y)/(1-p)$, and the second derivative is $-Y/p^2 - (1-Y)/(1-p)^2$. Taking expectations of the second derivative gives $-1/p - 1/(1-p) = -1/p(1-p)$. In other words, $V(p) = -1/\mathbb{E}[\ell'']$. Using weights inversely proportional to the variance thus turns out to be equivalent to dividing by the expected second derivative. But gradient divided by second derivative is the increment we use in Newton's method, QED.

11.4 Generalized Linear and Generalized Additive Models

Logistic regression is part of a broader family of **generalized linear models** (GLMs), where the conditional distribution of the response falls in some parametric family, and the parameters are set by the linear predictor. Ordinary, least-squares regression is the case where response is Gaussian, with mean equal to the linear predictor, and constant variance. Logistic regression is the case where the response is binomial, with n equal to the number of data-points with the given x (usually but not always 1), and p is given by Equation 11.5. Changing the relationship between the parameters and the linear predictor is called changing the **link function**. For computational reasons, the link function is actually the function you apply to the mean response to get back the linear predictor, rather than the other way around — (11.4) rather than (11.5). There are thus other forms of binomial regression besides logistic regression.[4] There is also Poisson regression (appropriate when the data are counts without any upper limit), gamma regression, etc.; we will say more about these in Chapter 12.

In R, any standard GLM can be fit using the (base) `glm` function, whose syntax is very similar to that of `lm`. The major wrinkle is that, of course, you need to specify the family of probability distributions to use, by the `family` option — `family=binomial` defaults to logistic regression. (See `help(glm)` for the gory details on how to do, say, probit regression.) All of these are fit by the same sort of numerical likelihood maximization.

11.4.1 Generalized Additive Models

A natural step beyond generalized linear models is **generalized additive models** (GAMs), where instead of making the transformed mean response a *linear* function of the inputs, we make it an *additive* function of the inputs. This means combining a function for fitting additive models with likelihood maximization. This is actually done in R with the same `gam` function we used for additive models (hence the name). We will look at how this works in some detail in Chapter 12. For now, the basic idea is that the iteratively re-weighted least squares procedure of §11.3.1 doesn't really require the model for the log odds to be linear. We get a GAM when we fit an additive model to the z_i; we could even fit an arbitrary non-parametric model, like a kernel regression, though that's not often done.

GAMs can be used to check GLMs in much the same way that smoothers can be used to check parametric regressions: fit a GAM and a GLM to the same data, then simulate from the GLM, and re-fit both models to the simulated data. Repeated many times, this gives a distribution for how much better the GAM will seem to fit than the GLM does, *even when the GLM is true*. You can then read a p-value off of this distribution. This is illustrated in §11.6 below.

[4] My experience is that these tend to give similar error rates as classifiers, but have rather different guesses about the underlying probabilities.

11.5 Model Checking

The validity of the logistic regression model is no more a fact of mathematics or nature than is the validity of the linear regression model. Both are sometimes convenient assumptions, but neither is guaranteed to be correct, nor even some sort of generally-correct default. In either case, if we want to use the model, the proper scientific (and statistical) procedure is to *check* the validity of the modeling assumptions.

11.5.1 Residuals

In your linear models course, you learned a lot of checks based on the residuals of the model (see Chapter 2). Many of these ideas translates to logistic regression, but we need to re-define residuals. Sometimes people work with the "response" residuals,

$$y_i - p(x_i) \tag{11.18}$$

which should have mean zero (why?), but are heteroskedastic even when the model is true (why?). Others work with standardized or **Pearson** residuals,

$$\frac{y_i - p(x_i)}{\sqrt{V(p(x_i))}} \tag{11.19}$$

and there are yet other notions of residuals for logistic models. Still, both the response and the Pearson residuals should be unpredictable from the covariates, and the latter should have constant variance.

11.5.2 Non-parametric Alternatives

Chapter 9 discussed how non-parametric regression models can be used to check whether parametric regressions are well-specified. The same ideas apply to logistic regressions, with the minor modification that in place of the difference in MSEs, one should use the difference in log-likelihoods, or (what comes to the same thing, up to a factor of 2) the difference in deviances. The use of generalized additive models (§11.4.1) as the alternative model class is illustrated in §11.6 below.

11.5.3 Calibration

Because logistic regression predicts actual *probabilities*, we can check its predictions in a more stringent way than an ordinary regression, which just tells us the mean value of Y, but is otherwise silent about its distribution. If we've got a model which tells us that the probability of rain on a certain class of days is 50%, it had better rain on half of those days, or there model is just *wrong* about the probability of rain. More generally, we'll say that the model is **calibrated** (or **well-calibrated**) when

$$\Pr\left(Y = 1 | \hat{p}(X) = p\right) = p \tag{11.20}$$

That is, the actual probabilities should match the predicted probabilities. If we have a large sample, by the law of large numbers, observed relative frequencies will converge on true probabilities. Thus, the observed relative frequencies should be close to the predicted probabilities, or else the model is making systematic mistakes.

In practice, each case often has its own unique predicted probability p, so we can't really accumulate many cases with the same p and check the relative frequency among those cases. When that happens, one option is to look at all the cases where the predicted probability is in some small range $[p, p + \epsilon)$; the observed relative frequency had then better be in that range too. §11.7 below illustrates some of the relevant calculations.

A second option is to use what is called a **proper scoring rule**, which is a function of the outcome variables and the predicted probabilities that attains its minimum when, and only when, the predicted are calibrated. For binary outcomes, one proper scoring rule (historically the oldest) is the **Brier score**,

$$n^{-1} \sum_{i=1}^{n} (y_i - p_i)^2 \tag{11.21}$$

Another however is simply the (normalized) negative log-likelihood,

$$-n^{-1} \sum_{i=1}^{n} y_i \log p_i + (1 - y_i) \log (1 - p_i) \tag{11.22}$$

Of course, proper scoring rules are better evaluated out-of-sample, or, failing that, through cross-validation, than in-sample. Even an in-sample evaluation is better than nothing, however, which is too often what happens.

11.6 A Toy Example

Here's a worked R example, using the data from the upper right panel of Figure 11.1. The 50×2 matrix `x` holds the input variables (the coordinates are independently and uniformly distributed on $[-1, 1]$), and `y.1` the corresponding class labels, themselves generated from a logistic regression with $\beta_0 = -0.5$, $\beta = (-1, 1)$.

```
df <- data.frame(y = y.1, x1 = x[, 1], x2 = x[, 2])
logr <- glm(y ~ x1 + x2, data = df, family = "binomial")
```

The **deviance** of a model fitted by maximum likelihood is twice the difference between its log likelihood and the maximum log likelihood for a **saturated** model, i.e., a model with one parameter per observation. Hopefully, the saturated model can give a perfect fit.[5] Here the saturated model would assign probability 1 to the observed outcomes[6], and the logarithm of 1 is zero, so $D = 2\ell(\widehat{\beta_0}, \widehat{\beta})$. The null deviance is what's achievable by using just a constant bias β_0 and setting the rest of β to 0. The fitted model definitely improves on that.[7]

If we're interested in inferential statistics on the estimated model, we can see those with `summary`, as with `lm`:

```
summary(logr, digits = 2, signif.stars = FALSE)
##
## Call:
## glm(formula = y ~ x1 + x2, family = "binomial", data = df)
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.34521  -0.82798   0.01499   0.83880   2.07197
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -0.5091     0.3793  -1.342  0.17957
## x1           -2.2365     0.7293  -3.067  0.00216 **
## x2            2.4894     0.8556   2.909  0.00362 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 69.315  on 49  degrees of freedom
## Residual deviance: 48.929  on 47  degrees of freedom
## AIC: 54.929
##
## Number of Fisher Scoring iterations: 5
```

[5] The factor of two is so that the deviance will have a χ^2 distribution. Specifically, if the model with p parameters is right, as $n \to \infty$ the deviance will approach a χ^2 distribution with $n - p$ degrees of freedom.

[6] This is not possible when there are multiple observations with the same input features, but different classes.

[7] AIC is of course the Akaike information criterion, $-2\ell + 2p$, with p being the number of parameters (here, $p = 3$). (Some people divide this through by n.) See §D.5.5.5 for more on AIC, and why I mostly ignore it in this book.

```
simulate.from.logr <- function(df, mdl) {
    probs <- predict(mdl, newdata = df, type = "response")
    df$y <- rbinom(n = nrow(df), size = 1, prob = probs)
    return(df)
}
```

CODE EXAMPLE 27: *Code for simulating from an* estimated *logistic regression model. By default (*`type="link"`*),* `predict` *for logistic regressions returns predictions for the log odds; changing the* `type` *to* `"response"` *returns a probability.*

The fitted values of the logistic regression are the class probabilities; this next line gives us the (in-sample) mis-classification rate.

```
mean(ifelse(fitted(logr) < 0.5, 0, 1) != df$y)
## [1] 0.16
```

An error rate of 16% may sound bad, but notice from the contour lines in Figure 11.1 that lots of the probabilities are near 0.5, meaning that the classes are just genuinely hard to predict.

To see how well the logistic regression assumption holds up, let's compare this to a GAM. We'll use the same package for estimating the GAM, `mgcv`, that we used to fit the additive models in Chapter 8.

```
library(mgcv)
(gam.1 <- gam(y ~ s(x1) + s(x2), data = df, family = "binomial"))
##
## Family: binomial
## Link function: logit
##
## Formula:
## y ~ s(x1) + s(x2)
##
## Estimated degrees of freedom:
## 1.00 2.09  total = 4.09
##
## UBRE score: 0.03774858
```

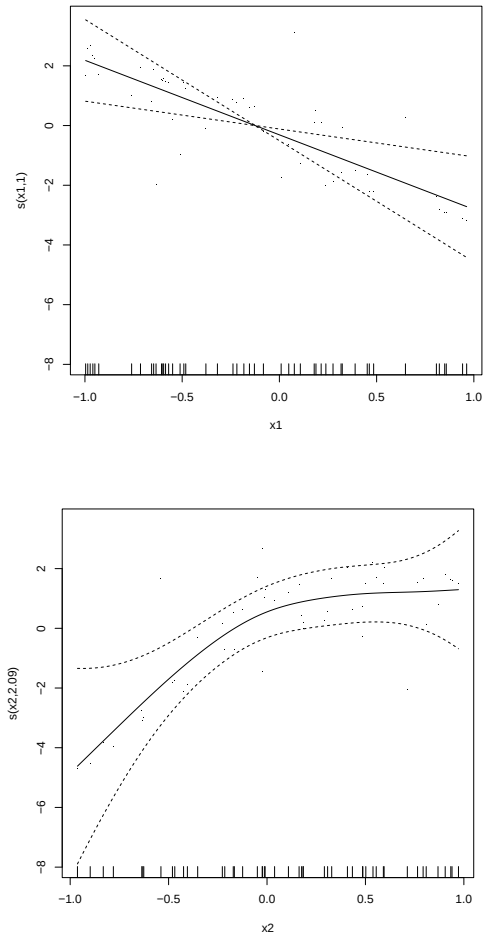
This fits a GAM to the same data, using spline smoothing of both input variables. (Figure 11.2 shows the partial response functions.) The (in-sample) deviance is

```
signif(gam.1$deviance, 3)
## [1] 43.7
```

which is lower than the logistic regression, so the GAM gives the data higher likelihood. We expect this; the question is whether the difference is significant, or within the range of what we should expect when logistic regression is valid. To test this, we need to simulate from the logistic regression model.

Now we simulate from our fitted model, and re-fit both the logistic regression and the GAM.

```
delta.deviance.sim <- function(df, mdl) {
    sim.df <- simulate.from.logr(df, mdl)
```




```
plot(gam.1, residuals = TRUE, pages = 0)
```

Figure 11.2 Partial response functions estimated when we fit a GAM to the data simulated from a logistic regression. Notice that the vertical axes are on the logit scale.

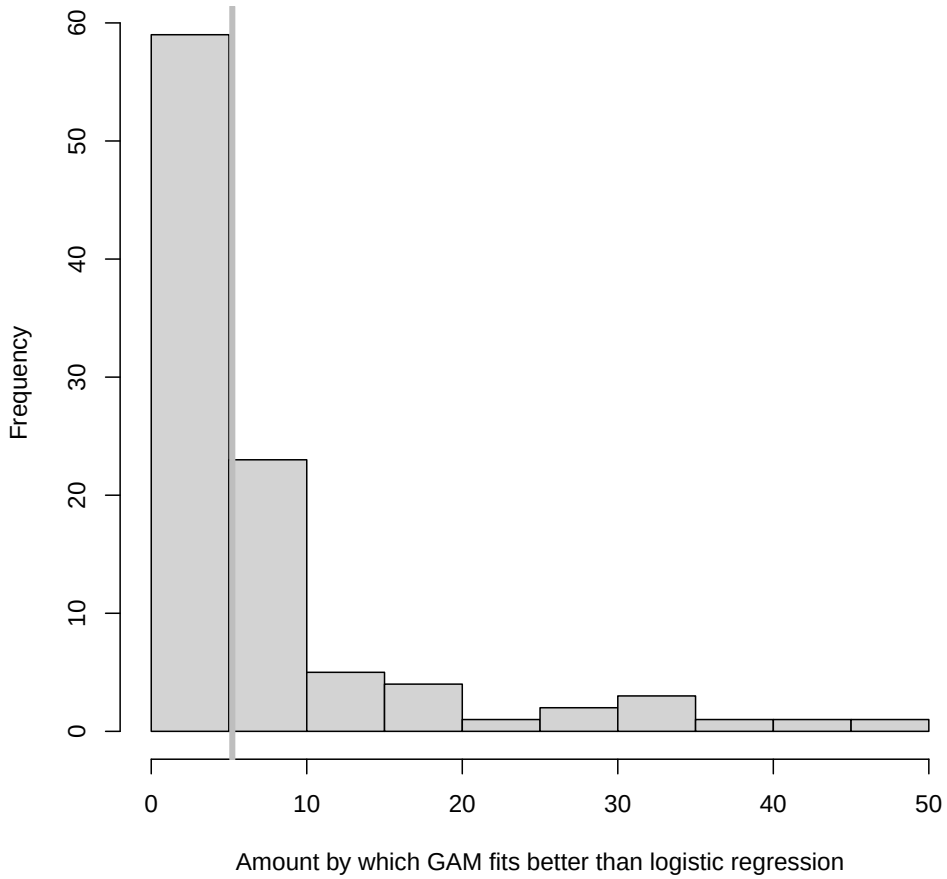
```
    GLM.dev <- glm(y ~ x1 + x2, data = sim.df, family = "binomial")$deviance
    GAM.dev <- gam(y ~ s(x1) + s(x2), data = sim.df, family = "binomial")$deviance
    return(GLM.dev - GAM.dev)
}
```

Notice that in this simulation we are not generating new $\vec{X}$ values. The logistic regression and the GAM are both models for the response *conditional* on the inputs, and are agnostic about how the inputs are distributed, or even whether it's meaningful to talk about their distribution.

Finally, we repeat the simulation a bunch of times, and see where the observed difference in deviances falls in the sampling distribution.

```
(delta.dev.observed <- logr$deviance - gam.1$deviance)
## [1] 5.212441
delta.dev <- replicate(100, delta.deviance.sim(df, logr))
mean(delta.dev.observed <= delta.dev)
## [1] 0.41
```

In other words, the amount by which a GAM fits the data better than logistic regression is pretty near the middle of the null distribution. Since the example data really *did* come from a logistic regression, this is a relief.




```
hist(delta.dev, main = "", xlab = "Amount by which GAM fits better than logistic regression")
abline(v = delta.dev.observed, col = "grey", lwd = 4)
```

Figure 11.3 Sampling distribution for the difference in deviance between a GAM and a logistic regression, on data generated from a logistic regression. The observed difference in deviances is shown by the grey vertical line.

```
snoqualmie <- scan("http://www.stat.washington.edu/peter/book.data/set1", skip = 1)
snoq <- data.frame(tomorrow = c(tail(snoqualmie, -1), NA), today = snoqualmie)
years <- 1948:1983
days.per.year <- rep(c(366, 365, 365, 365), length.out = length(years))
snoq$year <- rep(years, times = days.per.year)
snoq$day <- rep(c(1:366, 1:365, 1:365, 1:365), times = length(years)/4)
snoq <- snoq[-nrow(snoq), ]
```

Code Example 28: *Read in and re-shape the Snoqualmie data set. Prof. Guttorp, who has kindly provided the data, formatted it so that each year was a different row, which is rather inconvenient for R.*

11.7 Weather Forecasting in Snoqualmie Falls

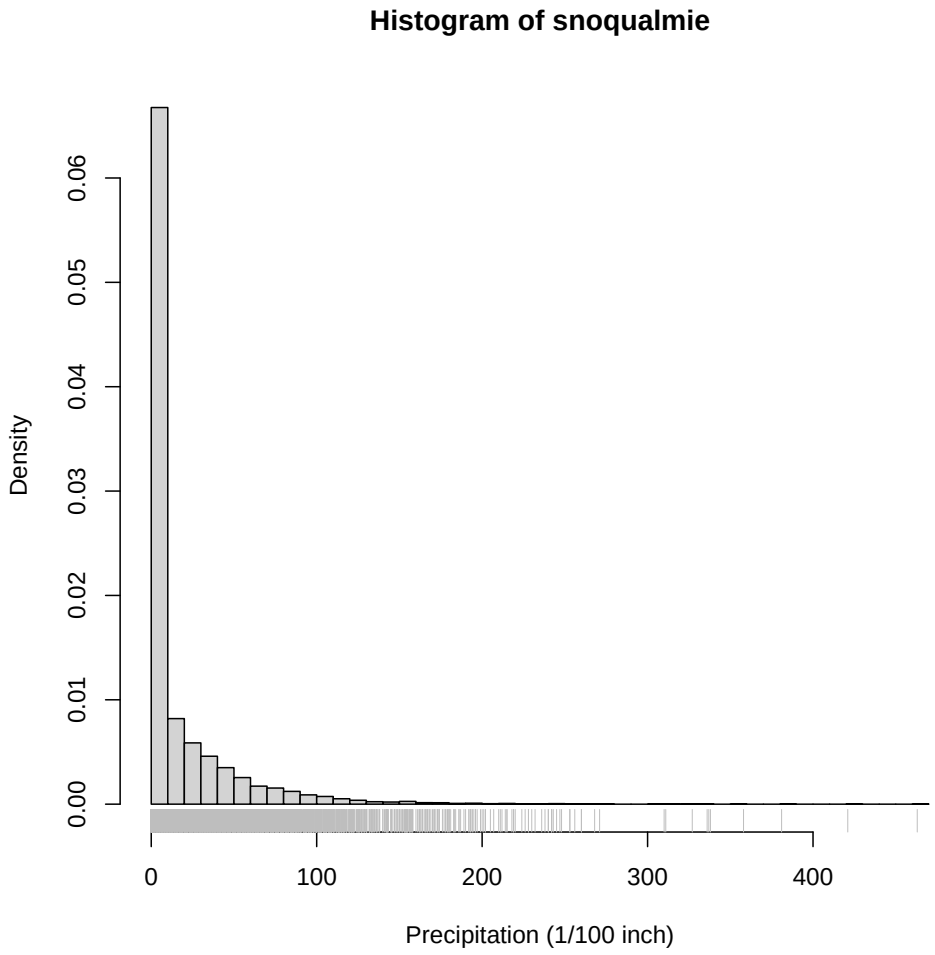
For our worked data example, we are going to build a simple weather forecaster. Our data consist of daily records, from the start of 1948 to the end of 1983, of precipitation at Snoqualmie Falls, Washington (Figure 11.4)[8]. Each row of the data file is a different year; each column records, for that day of the year, the day's precipitation (rain or snow), in units of $\frac{1}{100}$ inch. Because of leap-days, there are 366 columns, with the last column having an NA value for three out of four years.

What we want to do is predict tomorrow's weather from today's. This would be of interest if we lived in Snoqualmie Falls, or if we operated one of the local hydroelectric power plants, or the tourist attraction of the Falls themselves. Examining the distribution of the data (Figures 11.5 and 11.6) shows that there is a big spike in the distribution at zero precipitation, and that days of no precipitation can follow days of any amount of precipitation but seem to be less common after heavy precipitation.

[8] I learned of this data set from Guttorp (1995); the data file is available from `http://www.stat.washington.edu/peter/stoch.mod.data.html`. See Code Example 28 for the commands used to read it in, and to reshape it into a form more convenient for R.

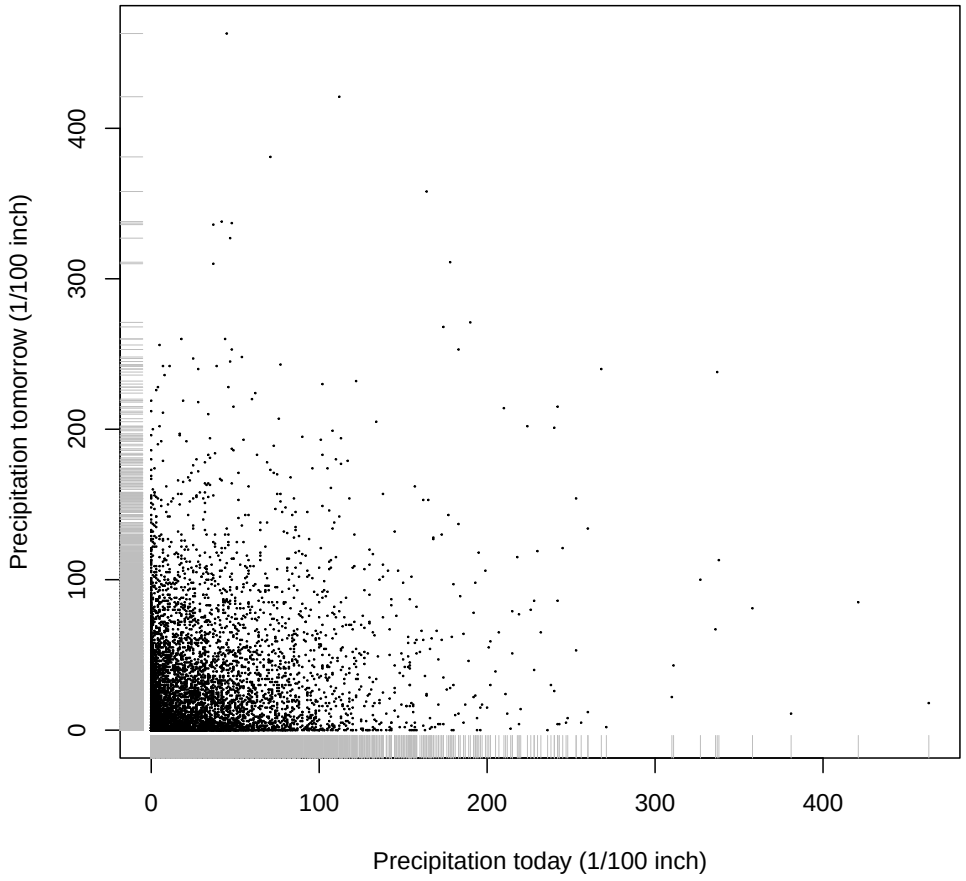


Figure 11.4 Snoqualmie Falls, Washington, on a low-precipitation day. Photo by Jeannine Hall Gailey, from `http://myblog.webbish6.com/2011/07/17-years-and-hoping-for-another-17.html`. [[TODO: Get permission for photo use!]]




```
hist(snoqualmie, n = 50, probability = TRUE, xlab = "Precipitation (1/100 inch)")
rug(snoqualmie, col = "grey")
```

Figure 11.5 Histogram of the amount of daily precipitation at Snoqualmie Falls




```
plot(tomorrow ~ today, data = snoq, xlab = "Precipitation today (1/100 inch)", ylab = "Precipitation
    cex = 0.1)
rug(snoq$today, side = 1, col = "grey")
rug(snoq$tomorrow, side = 2, col = "grey")
```

Figure 11.6 Scatterplot showing relationship between amount of precipitation on successive days. Notice that days of no precipitation can follow days of any amount of precipitation, but seem to be more common when there is little or no precipitation to start with.

These facts suggest that "no precipitation" is a special sort of event which would be worth predicting in its own right (as opposed to just being when the precipitation happens to be zero), so we will attempt to do so with logistic regression. Specifically, the input variable X_i will be the amount of precipitation on the ith day, and the response Y_i will be the indicator variable for whether there was any precipitation on day $i+1$ — that is, $Y_i = 1$ if $X_{i+1} > 0$, an $Y_i = 0$ if $X_{i+1} = 0$. We expect from Figure 11.6, as well as common experience, that the coefficient on X should be positive.[9]

The estimation is straightforward:

```
snoq.logistic <- glm((tomorrow > 0) ~ today, data = snoq, family = binomial)
```

To see what came from the fitting, run `summary`:

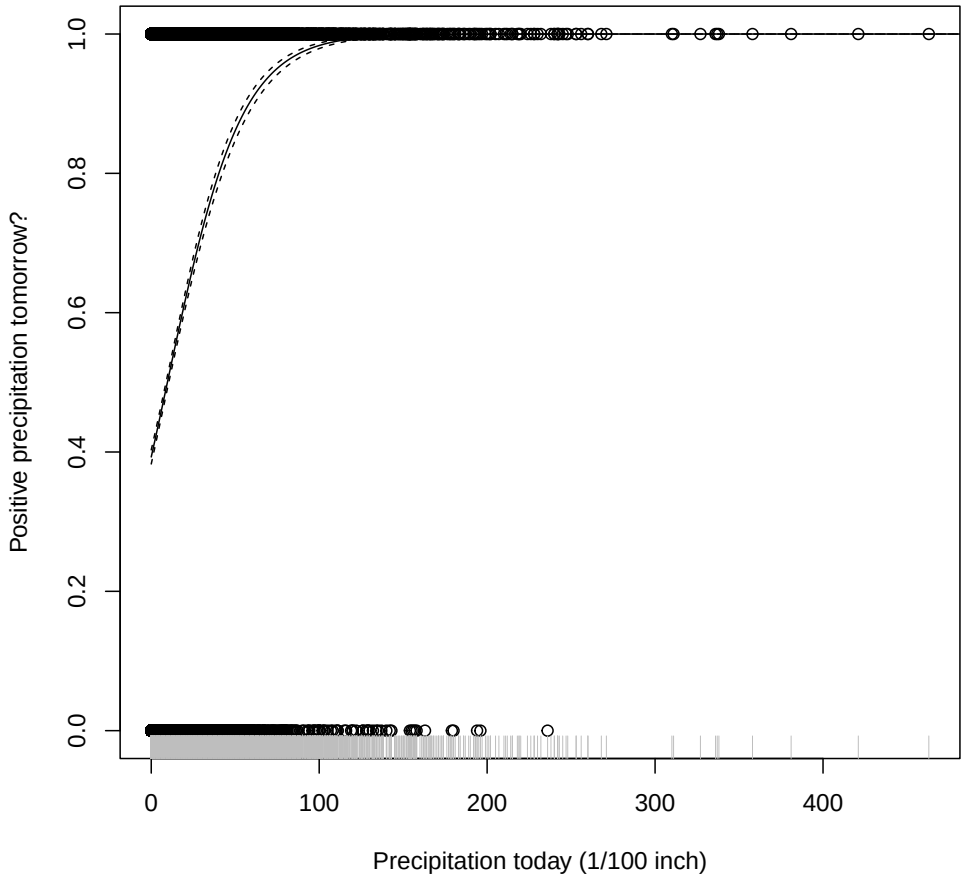
```
print(summary(snoq.logistic), digits = 3, signif.stars = FALSE)
##
## Call:
## glm(formula = (tomorrow > 0) ~ today, family = binomial, data = snoq)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -4.525  -0.999   0.167   1.170   1.367
##
## Coefficients:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.43520    0.02163   -20.1   <2e-16
## today        0.04523    0.00131    34.6   <2e-16
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 18191  on 13147  degrees of freedom
## Residual deviance: 15896  on 13146  degrees of freedom
## AIC: 15900
##
## Number of Fisher Scoring iterations: 5
```

The coefficient on the amount of precipitation today is indeed positive, and (if we can trust R's assumptions) highly significant. There is also an intercept term, which is slightly positive. We can see what the intercept term means by considering what happens on days of no precipitation. The linear predictor is then just the intercept, -0.435, and the predicted probability of precipitation is 0.393. That is, even when there is no precipitation today, it's almost as likely as not that there will be some precipitation tomorrow.[10]

We can get a more global view of what the model is doing by plotting the data and the predictions (Figure 11.7). This shows a steady increase in the probability of precipitation tomorrow as the precipitation today increases, though with the leveling off characteristic of logistic regression. The (approximate) 95% confidence limits for the predicted probability are (on close inspection) asymmetric.

[9] This does not attempt to model *how much* precipitation there will be tomorrow, if there is any. We could make that a separate model, if we can get this part right.

[10] For western Washington State, this is plausible — but see below.




```
plot((tomorrow > 0) ~ today, data = snoq, xlab = "Precipitation today (1/100 inch)",
    ylab = "Positive precipitation tomorrow?")
rug(snoq$today, side = 1, col = "grey")
data.plot <- data.frame(today = (0:500))
pred.bands <- function(mdl, data, col = "black", mult = 1.96) {
    preds <- predict(mdl, newdata = data, se.fit = TRUE)
    lines(data[, 1], ilogit(preds$fit), col = col)
    lines(data[, 1], ilogit(preds$fit + mult * preds$se.fit), col = col, lty = "dashed")
    lines(data[, 1], ilogit(preds$fit - mult * preds$se.fit), col = col, lty = "dashed")
}
pred.bands(snoq.logistic, data.plot)
```

Figure 11.7 Data (dots), plus predicted probabilities (solid line) and approximate 95% confidence intervals from the logistic regression model (dashed lines). Note that calculating standard errors for predictions on the logit scale, and then transforming, is better practice than getting standard errors directly on the probability scale.

How well does this work? We can get a first sense of this by comparing it to a simple nonparametric smoothing of the data. Remembering that when Y is binary, $\Pr(Y = 1|X = x) = \mathbb{E}[Y|X = x]$, we can use a smoothing spline to estimate $\mathbb{E}[Y|X = x]$ (Figure 11.8). This would not be so great as a model — it ignores the fact that the response is a binary event and we're trying to estimate a probability, the fact that the variance of Y therefore depends on its mean, etc. — but it's at least suggestive.

The result starts out notably above the logistic regression, then levels out and climbs much more slowly. It also has a bunch of dubious-looking wiggles, despite the cross-validation.

We can try to do better by fitting a generalized additive model. In this case, with only one predictor variable, this means using non-parametric smoothing to estimate the log odds — we're still using the logistic transformation, but only requiring that the log odds change smoothly with X, not that they be linear in X. The result (Figure 11.9) is initially similar to the spline, but has some more exaggerated undulations, and has confidence intervals. At the largest values of X, the latter span nearly the whole range from 0 to 1, which is not unreasonable considering the sheer lack of data there.

Visually, the logistic regression curve is hardly ever within the confidence limits of the non-parametric predictor. What can we say about the difference between the two models more quantiatively?

Numerically, the deviance is 1.59×10^4 for the logistic regression, and 1.51×10^4 for the GAM. We can go through the testing procedure outlined in §11.6. We need a simulator (which presumes that the logistic regression model is true), and we need to calculate the difference in deviance on simulated data many times.

```
snoq.sim <- function(model = snoq.logistic) {
    fitted.probs = fitted(model)
    return(rbinom(n = length(fitted.probs), size = 1, prob = fitted.probs))
}
```

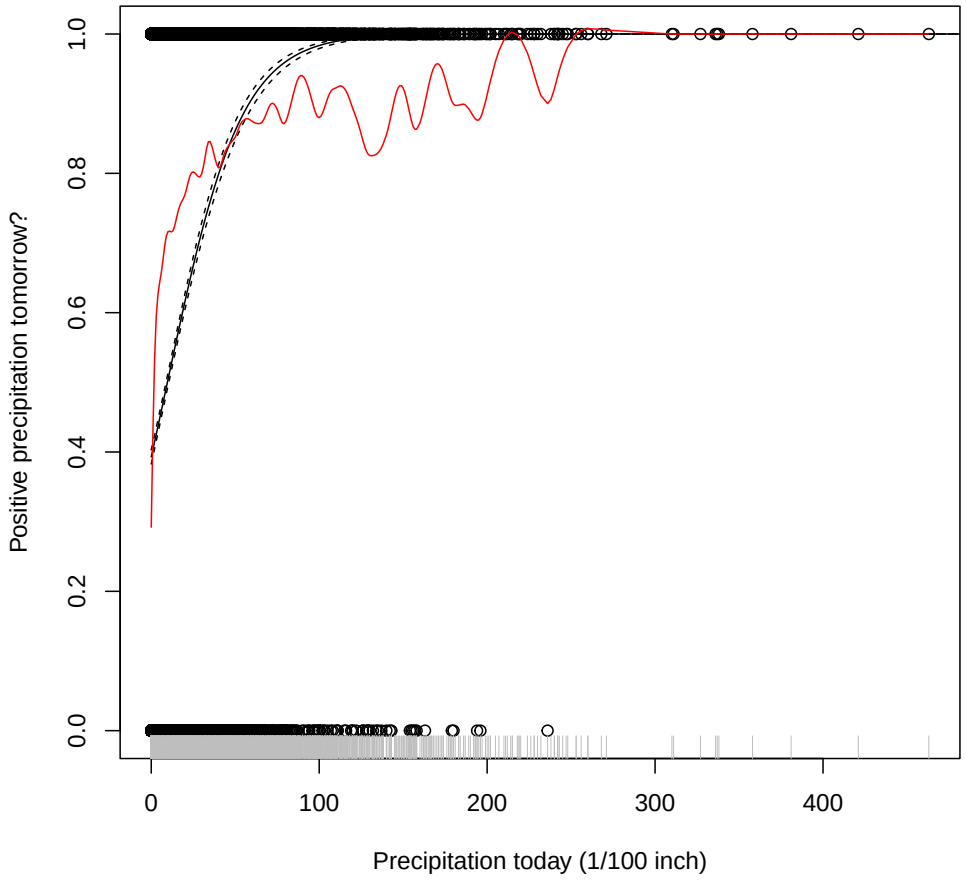
A quick check of the simulator against the observed values:

```
summary(ifelse(snoq[, 1] > 0, 1, 0))
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5262  1.0000  1.0000
summary(snoq.sim())
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  1.0000  0.5241  1.0000  1.0000
```

This suggests that the simulator is not acting crazily.

Now for the difference in deviances:

```
diff.dev <- function(model = snoq.logistic, x = snoq[, "today"]) {
    y.new <- snoq.sim(model)
    GLM.dev <- glm(y.new ~ x, family = binomial)$deviance
    GAM.dev <- gam(y.new ~ s(x), family = binomial)$deviance
    return(GLM.dev - GAM.dev)
}
```




```
plot((tomorrow > 0) ~ today, data = snoq, xlab = "Precipitation today (1/100 inch)",
    ylab = "Positive precipitation tomorrow?")
rug(snoq$today, side = 1, col = "grey")
data.plot <- data.frame(today = (0:500))
pred.bands(snoq.logistic, data.plot)
snoq.spline <- smooth.spline(x = snoq$today, y = (snoq$tomorrow > 0))
lines(snoq.spline, col = "red")
```

Figure 11.8 As Figure 11.7, plus a smoothing spline (red).

A single run of this takes about 0.6 seconds on my computer.

Finally, we calculate the distribution of difference in deviances under the null (that the logistic regression is properly specified), and the corresponding p-value:

```
diff.dev.obs <- snoq.logistic$deviance - snoq.gam$deviance
null.dist.of.diff.dev <- replicate(100, diff.dev())
p.value <- (1 + sum(null.dist.of.diff.dev > diff.dev.obs))/(1 + length(null.dist.of.diff.dev))
```

Using a thousand replicates takes about 67 seconds, or a bit over a minute; it gives a p-value of $< 1/101$. (A longer run of 1000 replicates, not shown, gives a p-values of $< 10^{-3}$.)

Having detected that there is a problem with the logistic model, we can ask where it lies. We *could* just use the GAM, but it's more interesting to try to diagnose what's going on.

In this respect Figure 11.9 is actually a little misleading, because it leads the eye to emphasize the disagreement between the models at large X, when actually there are very few data points there, and so even large differences in predicted probabilities there contribute little to the over-all likelihood difference. What is actually more important is what happens at $X = 0$, which contains a very large number of observations (about 47% of all observations), and which we have reason to think is a special value anyway.

Let's try introducing a dummy variable for $X = 0$ into the logistic regression, and see what happens. It will be convenient to augment the data frame with an extra column, recording 1 whenever $X = 0$ and 0 otherwise.

```
snoq2 <- data.frame(snoq, dry = ifelse(snoq$today == 0, 1, 0))
snoq2.logistic <- glm((tomorrow > 0) ~ today + dry, data = snoq2, family = binomial)
snoq2.gam <- gam((tomorrow > 0) ~ s(today) + dry, data = snoq2, family = binomial)
```

Notice that I allow the GAM to treat zero as a special value as well, by giving it access to that dummy variable. In principle, with enough data it can decide whether or not that is useful on its own, but since we have guessed that it is, we might as well include it. The new GLM has a deviance of 1.5×10^4, lower than even the GAM before, and the new GAM has a deviance of 1.48×10^4. I will leave repeating the specification test as an exercise. Figure 11.10 shows the data and the two new models. These are *extremely* close to each other at low percipitation, and diverge thereafter. The new GAM is the smoothest model we've seen yet, which suggests that before the it was being under-smoothed to help capture the special value at zero.

Let's turn now to looking at calibration. The actual fraction of no-precipitation days which are followed by precipitation is

```
signif(mean(snoq$tomorrow[snoq$today == 0] > 0), 3)
## [1] 0.287
```

What does the new logistic model predict?

```
signif(predict(snoq2.logistic, newdata = data.frame(today = 0, dry = 1), type = "response"),
    3)
##     1
## 0.287
```

This should not be surprising — we've given the model a special parameter

dedicated to getting this one probability exactly right! The hope however is that this will change the predictions made on days *with* precipitation so that they are better.

Looking at a histogram of fitted values (`hist(fitted(snoq2.logistic))`) shows a gap in the distribution of predicted probabilities below 0.63, so we'll look first at days where the predicted probability is between 0.63 and 0.64.

```
signif(mean(snoq$tomorrow[(fitted(snoq2.logistic) >= 0.63) & (fitted(snoq2.logistic) <
    0.64)] > 0), 3)
## [1] 0.526
```

Not bad — but a bit painful to write out. Let's write a function:

```
frequency.vs.probability <- function(p.lower, p.upper = p.lower + 0.01, model = snoq2.logistic,
    events = (snoq$tomorrow > 0)) {
    fitted.probs <- fitted(model)
    indices <- (fitted.probs >= p.lower) & (fitted.probs < p.upper)
    ave.prob <- mean(fitted.probs[indices])
    frequency <- mean(events[indices])
    se <- sqrt(ave.prob * (1 - ave.prob)/sum(indices))
    return(c(frequency = frequency, ave.prob = ave.prob, se = se))
}
```

I have added a calculation of the average predicted probability, and a crude estimate of the standard error we should expect if the observations really are binomial with the predicted probabilities[11]. Try the function out before doing anything rash:

```
frequency.vs.probability(0.63)
##  frequency   ave.prob         se
## 0.52603037 0.63414568 0.01586292
```

This agrees with our previous calculation.

Now we can do this for a lot of probability brackets:

```
f.vs.p <- sapply(c(0.28, (63:100)/100), frequency.vs.probability)
```

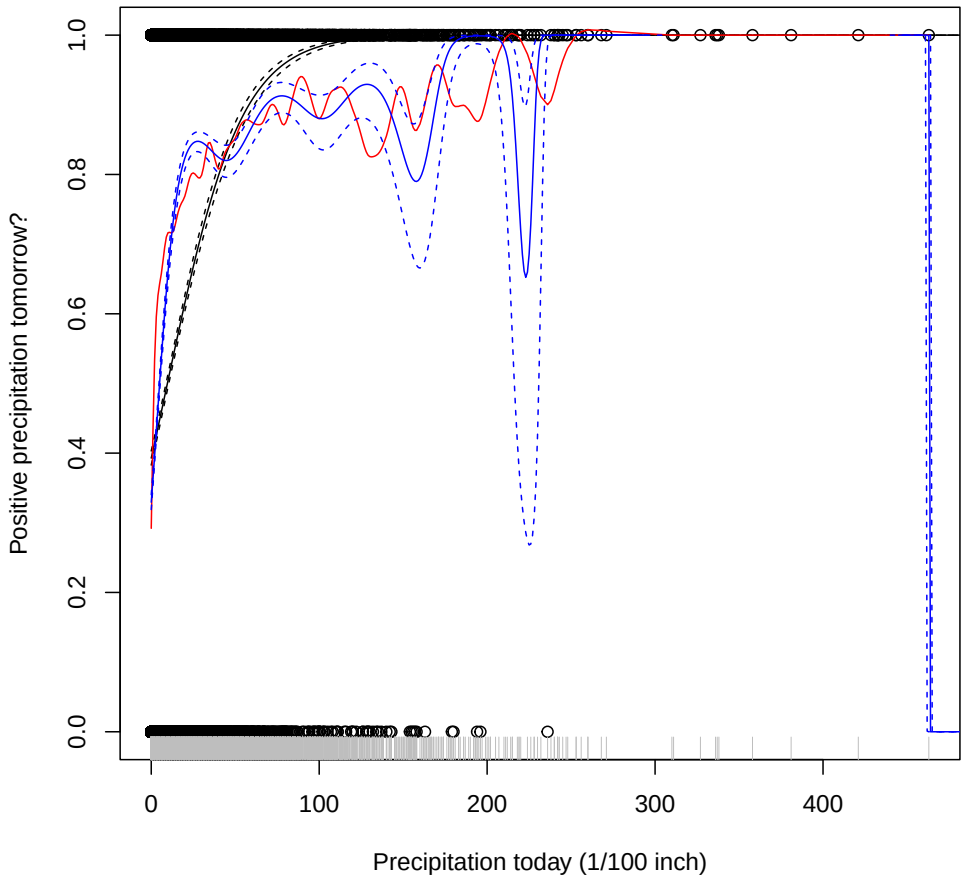
This comes with some unfortunate R cruft, removable thus

```
f.vs.p <- data.frame(frequency = f.vs.p["frequency", ], ave.prob = f.vs.p["ave.prob",
    ], se = f.vs.p["se", ])
```

and we're ready to plot (Figure 11.11). The observed frequencies are generally reasonably near the predicted probabilities. While I wouldn't want to say this was the last word in weather forecasting[12], it's surprisingly good for such a simple model. I will leave calibration checking for the GAM as another exercise.

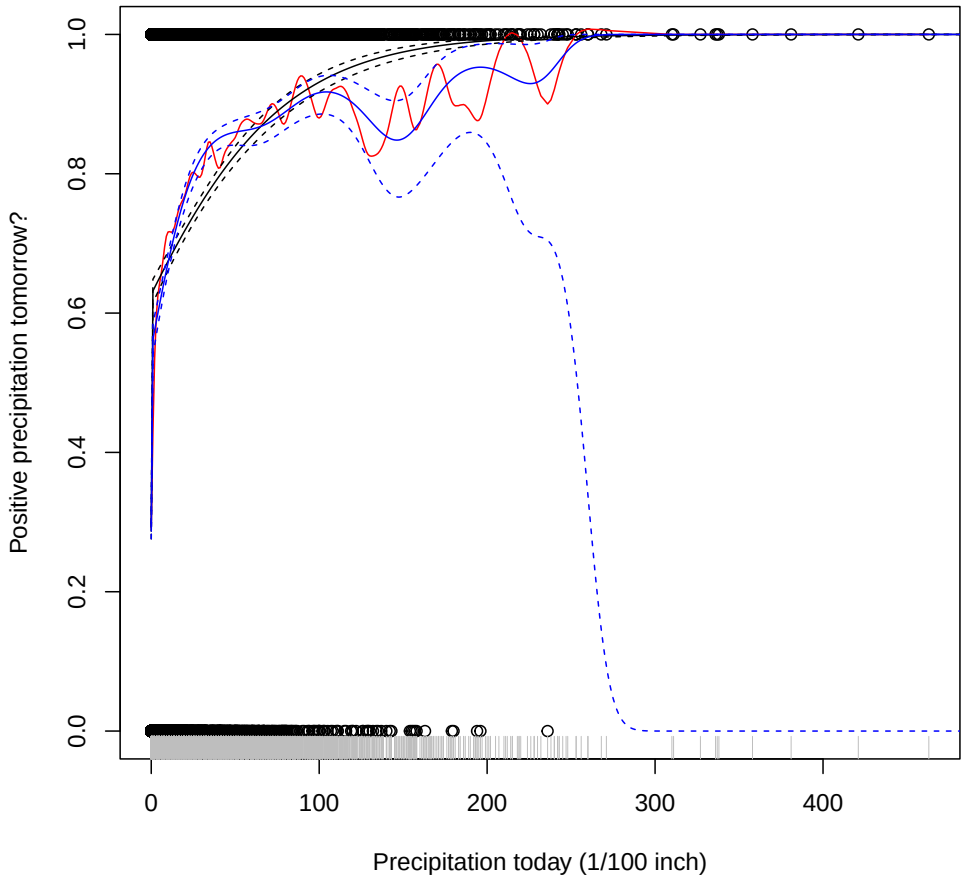
[11] This could be improved by averaging predicted variances for each point, but using probability ranges of 0.01 makes it hardly worth the effort.

[12] There is an extensive discussion of this data in Guttorp (1995, ch. 2), including many significant refinements, such as dependence across multiple days.



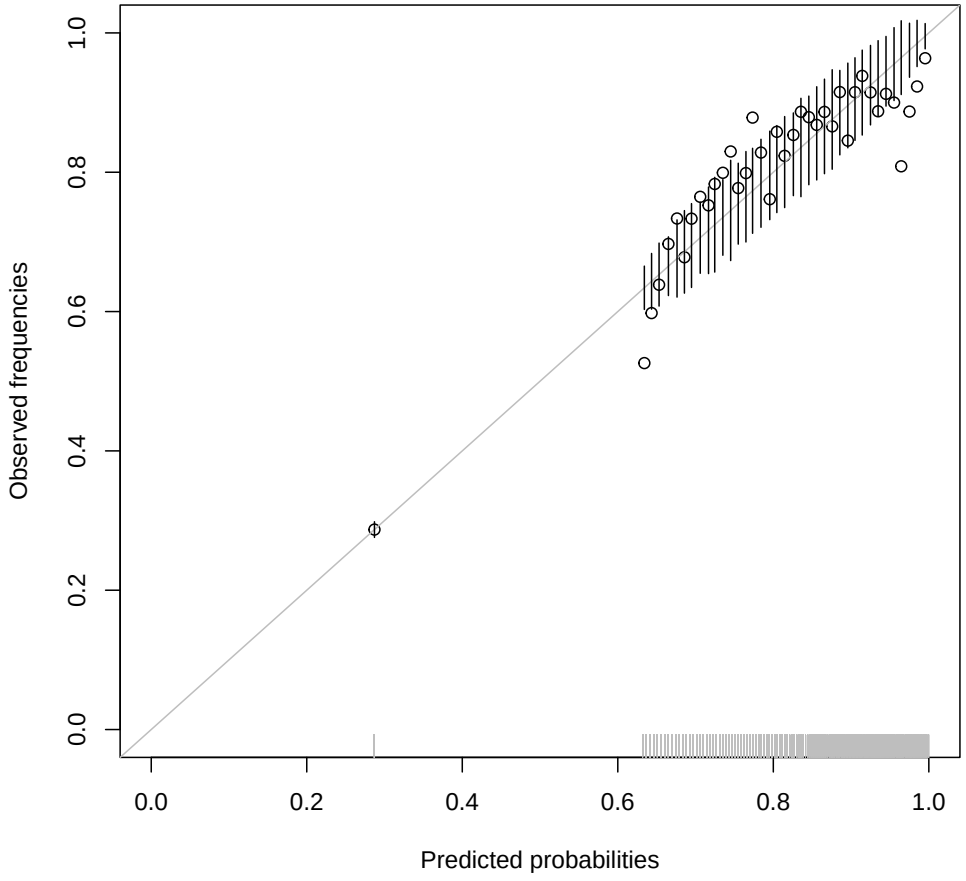

```
library(mgcv)
plot((tomorrow > 0) ~ today, data = snoq, xlab = "Precipitation today (1/100 inch)",
    ylab = "Positive precipitation tomorrow?")
rug(snoq$today, side = 1, col = "grey")
pred.bands(snoq.logistic, data.plot)
lines(snoq.spline, col = "red")
snoq.gam <- gam((tomorrow > 0) ~ s(today), data = snoq, family = binomial)
pred.bands(snoq.gam, data.plot, "blue")
```

Figure 11.9 As Figure 11.8, but with the addition of a generalized additive model (blue line) and its confidence limits (dashed blue lines).




```
plot((tomorrow > 0) ~ today, data = snoq, xlab = "Precipitation today (1/100 inch)",
    ylab = "Positive precipitation tomorrow?")
rug(snoq$today, side = 1, col = "grey")
data.plot = data.frame(data.plot, dry = ifelse(data.plot$today == 0, 1, 0))
lines(snoq.spline, col = "red")
pred.bands(snoq2.logistic, data.plot)
pred.bands(snoq2.gam, data.plot, "blue")
```

Figure 11.10 As Figure 11.9, but allowing the two models to use a dummy variable indicating when today is completely dry ($X = 0$).




```
plot(frequency ~ ave.prob, data = f.vs.p, xlim = c(0, 1), ylim = c(0, 1), xlab = "Predicted probabil
    ylab = "Observed frequencies")
rug(fitted(snoq2.logistic), col = "grey")
abline(0, 1, col = "grey")
segments(x0 = f.vs.p$ave.prob, y0 = f.vs.p$ave.prob - 1.96 * f.vs.p$se, y1 = f.vs.p$ave.prob +
    1.96 * f.vs.p$se)
```

Figure 11.11 Calibration plot for the modified logistic regression model `snoq2.logistic`. Points show the actual frequency of precipitation for each level of predicted probability. Vertical lines are (approximate) 95% sampling intervals for the frequency, given the predicted probability and the number of observations.

11.8 Logistic Regression with More Than Two Classes

If Y can take on more than two values, say k of them, we can still use logistic regression. Instead of having one set of parameters β_0, β, each class c in $0 : (k-1)$ will have its own offset $\beta_0^{(c)}$ and vector $\beta^{(c)}$, and the predicted conditional probabilities will be

$$\Pr\left(Y = c|\vec{X} = x\right) = \frac{e^{\beta_0^{(c)} + x \cdot \beta^{(c)}}}{\sum_c e^{\beta_0^{(c)} + x \cdot \beta^{(c)}}} \tag{11.23}$$

You can check that when there are only two classes (say, 0 and 1), equation 11.23 reduces to equation 11.5, with $\beta_0 = \beta_0^{(1)} - \beta_0^{(0)}$ and $\beta = \beta^{(1)} - \beta^{(0)}$. In fact, no matter how many classes there are, we can always pick one of them, say $c = 0$, and fix its parameters at exactly zero, without any loss of generality (Exercise 11.2)[13].

Calculation of the likelihood now proceeds as before (only with more bookkeeping), and so does maximum likelihood estimation.

As for R implementations, for am long time the easiest way to do this was actually to use the `nnet` package for neural networks (Venables and Ripley, 2002). More recently, the `multiclass` function from the `mgcv` package does the same sort of job, with an interface closer to what you will be familiar with from `glm` and `gam`.

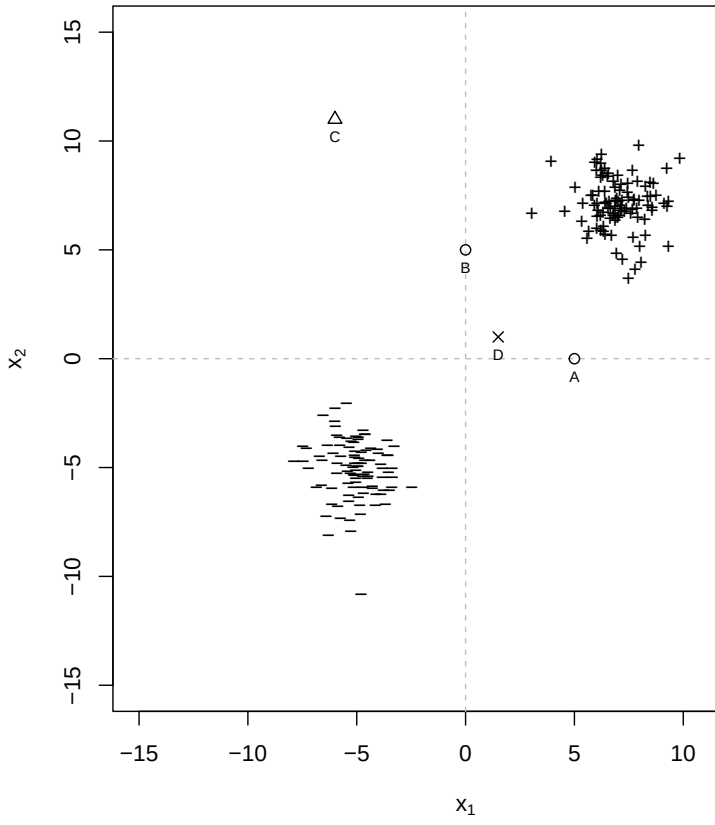
[13] Since we can arbitrarily chose which class's parameters to "zero out" without affecting the predicted probabilities, strictly speaking the model in Eq. 11.23 is **unidentified**. That is, different parameter settings lead to *exactly* the same outcome, so we can't use the data to tell which one is right. The usual response here is to deal with this by a convention: we *decide* to zero out the parameters of the first class, and then estimate the contrasting parameters for the others.

Exercises

11.1 *"We minimize the mis-classification rate by predicting the most likely class"*: Let $\widehat{\mu}(x)$ be our predicted class, either 0 or 1. Our error rate is then $\Pr(Y \neq \widehat{\mu})$. Show that $\Pr(Y \neq \widehat{\mu}) = \mathbb{E}\left[(Y - \widehat{\mu})^2\right]$. Further show that $\mathbb{E}\left[(Y - \widehat{\mu})^2 \mid X = x\right] = \Pr(Y = 1|X = x)\,(1 - 2\widehat{\mu}(x)) + \widehat{\mu}^2(x)$. Conclude by showing that if $\Pr(Y = 1|X = x) > 0.5$, the risk of mis-classification is minimized by taking $\widehat{\mu} = 1$, that if $\Pr(Y = 1|X = x) < 0.5$ the risk is minimized by taking $\widehat{\mu} = 0$, and that when $\Pr(Y = 1|X = x) = 0.5$ both predictions are equally risky.

11.2 A multiclass logistic regression, as in Eq. 11.23, has parameters $\beta_0^{(c)}$ and $\beta^{(c)}$ for each class c. Show that we can always get the same predicted probabilities by setting $\beta_0^{(c)} = 0$, $\beta^{(c)} = 0$ for any one class c, and adjusting the parameters for the other classes appropriately.

11.3 Find the first and second derivatives of the log-likelihood for logistic regression with one predictor variable. Explicitly write out the formula for doing one step of Newton's method. Explain how this relates to re-weighted least squares.



11.4 *Intuition-building with logistic regression*

In a classification problem, we get data like that shown above, with the two classes indicated by whether a point is plotted with + or −. We decide to use a logistic regression model. We estimate $\hat{\beta}_0 = 0$, $\hat{\beta}_1 = 1$ and $\hat{\beta}_2 = 1$.

1. Draw the line where the log-odds = 0, and so where $p = 1/2$. Does this separate most of the + points from the − points? Does it separate *all* of them?
2. Explain why the points marked A and B will have exactly the same probability of $Y = 1$, and why that probability will be $> 1/2$.
3. Explain why the point marked C will have nearly the same probability for $Y = 1$ as A and B do, while point D will have a much lower probability, even though D is much closer than C to both A and B.

12

Generalized Linear Models and Generalized Additive Models

12.1 Generalized Linear Models and Iterative Least Squares

Logistic regression is a particular instance of a broader kind of model, called a **generalized linear model** (GLM). You are familiar, of course, from your regression class with the idea of transforming the response variable, what we've been calling Y, and then predicting the transformed variable from X. This was *not* what we did in logistic regression. Rather, we transformed the *conditional expected value*, and made *that* a linear function of X. This seems odd, because it *is* odd, but it turns out to be useful.

Let's be specific. Our usual focus in regression modeling has been the conditional expectation function, $\mu(x) = \mathbb{E}[Y|X = x]$. In plain linear regression, we try to approximate $\mu(x)$ by $\beta_0 + x \cdot \beta$. In logistic regression, $\mu(x) = \mathbb{E}[Y|X = x] = \Pr(Y = 1|X = x)$, and it is a transformation of $\mu(x)$ which is linear. The usual notation says

$$\eta(x) = \beta_0 + x \cdot \beta \tag{12.1}$$

$$\eta(x) = \log \frac{\mu(x)}{1 - \mu(x)} \tag{12.2}$$

$$= g(\mu(x)) \tag{12.3}$$

defining the logistic **link function** by $g(m) = \log m/(1 - m)$. The function $\eta(x)$ is called the **linear predictor**.

Now, the first impulse for estimating this model would be to apply the transformation g to the response. But Y is always zero or one, so $g(Y) = \pm\infty$, and regression will not be helpful here. The standard strategy is instead to use (what else?) Taylor expansion. Specifically, we try expanding $g(Y)$ around $\mu(x)$, and stop at first order:

$$g(Y) \approx g(\mu(x)) + (Y - \mu(x))g'(\mu(x)) \tag{12.4}$$

$$= \eta(x) + (Y - \mu(x))g'(\mu(x)) \equiv z \tag{12.5}$$

We *define* this to be our effective response after transformation. Notice that if there were no noise, so that y was always equal to its conditional mean $\mu(x)$, then regressing z on x would give us back exactly the coefficients β_0, β. What this suggests is that we can *estimate* those parameters by regressing z on x.

The term $Y - \mu(x)$ has expectation zero, so it acts like the noise, with the factor of g' telling us about how the noise is scaled by the transformation. This

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

lets us work out the variance of z:

$$\mathbb{V}[Z|X=x] = \mathbb{V}[\eta(x)|X=x] + \mathbb{V}[(Y-\mu(x))g'(\mu(x))|X=x] \quad (12.6)$$
$$= 0 + (g'(\mu(x)))^2 \mathbb{V}[Y|X=x] \quad (12.7)$$

For logistic regression, with Y binary, $\mathbb{V}[Y|X=x] = \mu(x)(1-\mu(x))$. On the other hand, with the logistic link function, $g'(\mu(x)) = \frac{1}{\mu(x)(1-\mu(x))}$. Thus, for logistic regression, $\mathbb{V}[Z|X=x] = [\mu(x)(1-\mu(x))]^{-1}$.

Because the variance of Z changes with X, this is a heteroskedastic regression problem. As we saw in chapter 10, the appropriate way of dealing with such a problem is to use weighted least squares, with weights inversely proportional to the variances. This means that, in logistic regression, the weight at x should be proportional to $\mu(x)(1-\mu(x))$. Notice two things about this. First, the weights depend on the current guess about the parameters. Second, we give lots of weight to cases where $\mu(x) \approx 0$ or where $\mu(x) \approx 1$, and little weight to those where $\mu(x) = 0.5$. This focuses our attention on places where we have a lot of potential information — the distinction between a probability of 0.499 and 0.501 is just a lot harder to discern than that between 0.001 and 0.003!

We can now put all this together into an estimation strategy for logistic regression.

1. Get the data $(x_1, y_1), \ldots (x_n, y_n)$, and some initial guesses β_0, β.
2. until β_0, β converge
   1. Calculate $\eta(x_i) = \beta_0 + x_i \cdot \beta$ and the corresponding $\widehat{\mu}(x_i)$
   2. Find the effective transformed responses $z_i = \eta(x_i) + \frac{y_i - \widehat{\mu}(x_i)}{\widehat{\mu}(x_i)(1-\widehat{\mu}(x_i))}$
   3. Calculate the weights $w_i = \widehat{\mu}(x_i)(1-\widehat{\mu}(x_i))$
   4. Do a weighted linear regression of z_i on x_i with weights w_i, and set β_0, β to the intercept and slopes of this regression

Our initial guess about the parameters tells us about the heteroskedasticity, which we use to improve our guess about the parameters, which we use to improve our guess about the variance, and so on, until the parameters stabilize. This is called **iterative reweighted least squares** (or "iterative weighted least squares", "iteratively weighted least squares", "iterated reweighted least squares", etc.), abbreviated IRLS, IRWLS, IWLS, etc. As mentioned in the last chapter, this turns out to be *almost* equivalent to Newton's method, at least for this problem.

12.1.1 GLMs in General

The set-up for an arbitrary GLM is a generalization of that for logistic regression. We need

- A **linear predictor**, $\eta(x) = \beta_0 + x \cdot \beta$
- A **link function** g, so that $\eta(x) = g(\mu(x))$. For logistic regression, we had $g(\mu) = \log \mu/(1-\mu)$.

- A **dispersion scale function** V, so that $\mathbb{V}[Y|X=x] = \sigma^2 V(\mu(x))$. For logistic regression, we had $V(\mu) = \mu(1-\mu)$, and $\sigma^2 = 1$.

With these, we know the conditional mean and conditional variance of the response for each value of the input variables x.

As for estimation, basically everything in the IRWLS set up carries over unchanged. In fact, we can go through this algorithm:

1. Get the data $(x_1, y_1), \ldots (x_n, y_n)$, fix link function $g(\mu)$ and dispersion scale function $V(\mu)$, and make some initial guesses β_0, β.
2. Until β_0, β converge:
    1. Calculate $\eta(x_i) = \beta_0 + x_i \cdot \beta$ and the corresponding $\widehat{\mu}(x_i)$
    2. Find the effective transformed responses $z_i = \eta(x_i) + (y_i - \widehat{\mu}(x_i))g'(\widehat{\mu}(x_i))$
    3. Calculate the weights $w_i = [(g'(\widehat{\mu}(x_i))^2 V(\widehat{\mu}(x_i))]^{-1}$
    4. Do a weighted linear regression of z_i on x_i with weights w_i, and set β_0, β to the intercept and slopes of this regression

Notice that even if we don't know the over-all variance scale σ^2, that's OK, because the weights just have to be *proportional* to the inverse variance.

12.1.2 Examples of GLMs

12.1.2.1 Vanilla Linear Models

To re-assure ourselves that we are not doing anything crazy, let's see what happens when $g(\mu) = \mu$ (the "identity link"), and $\mathbb{V}[Y|X=x] = \sigma^2$, so that $V(\mu) = 1$. Then $g' = 1$, all weights $w_i = 1$, and the effective transformed response $z_i = y_i$. So we just end up regressing y_i on x_i with no weighting at all — we do ordinary least squares. Since neither the weights nor the transformed response will change, IRWLS will converge exactly after one step. So if we get rid of all this nonlinearity and heteroskedasticity and go all the way back to our very first days of doing regression, we get the OLS answers we know and love.

12.1.2.2 Binomial Regression

In many situations, our response variable y_i will be an integer count running between 0 and some pre-determined upper limit n_i. (Think: number of patients in a hospital ward with some condition, number of children in a classroom passing a test, number of widgets produced by a factory which are defective, number of people in a village with some genetic mutation.) One way to model this would be as a binomial random variable, with n_i trials, and a success probability p_i which is a logistic function of predictors x. The logistic regression we have done so far is the special case where $n_i = 1$ always. I will leave it as an Exercise (12.1) for you to work out the link function and the weights for general binomial regression, where the n_i are treated as known.

One implication of this model is that each of the n_i "trials" aggregated together in y_i is independent of all the others, at least once we condition on the predictors

x. (So, e.g., whether any student passes the test is independent of whether any of their classmates pass, once we have conditioned on, say, teacher quality and average previous knowledge.) This may or may not be a reasonable assumption. When the successes or failures are dependent, even after conditioning on the predictors, the binomial model will be mis-specified. We can either try to get more information, and hope that conditioning on a richer set of predictors makes the dependence go away, or we can just try to account for the dependence by modifying the variance ("overdispersion" or "underdispersion"); we'll return to both topics in §12.1.4.

12.1.2.3 Poisson Regression

Recall that the Poisson distribution has probability mass function

$$p(y) = \frac{e^{-\mu}\mu^y}{y!} \tag{12.8}$$

with $\mathbb{E}[Y] = \mathbb{V}[Y] = \mu$. As you remember from basic probability, a Poisson distribution is what we get from a binomial if the probability of success per trial shrinks towards zero but the number of trials grows to infinity, so that we keep the mean number of successes the same:

$$\text{Binom}(n, \mu/n) \rightsquigarrow \text{Pois}(\mu) \tag{12.9}$$

This makes the Poisson distribution suitable for modeling counts with no fixed upper limit, but where the probability that any one of the many individual trials is a success is fairly low. If μ is allowed to change with the predictor variables, we get Poisson regression. Since the variance is equal to the mean, Poisson regression is always going to be heteroskedastic.

Since μ has to be non-negative, a natural link function is $g(\mu) = \log\mu$. This produces $g'(\mu) = 1/\mu$, and so weights $w = \mu$. When the expected count is large, so is the variance, which normally would reduce the weight put on an observation in regression, but in this case large expected counts also provide more information about the coefficients, so they end up getting increasing weight.

12.1.3 Uncertainty

Standard errors for coefficients can be worked out as in the case of weighted least squares for linear regression. Confidence intervals for the coefficients will be approximately Gaussian in large samples, for the usual likelihood-theory reasons, when the model is properly specified. One can, of course, also use either a parametric bootstrap, or resampling of cases/data-points to assess uncertainty.

Resampling of residuals can be trickier, because it is not so clear what counts as a residual. When the response variable is continuous, we can get "standardized" or "Pearson" residuals, $\hat{\epsilon}_i = \frac{y_i - \widehat{\mu}(x_i)}{\sqrt{\widehat{V(\mu(x_i))}}}$, resample them to get $\tilde{\epsilon}_i$, and then add $\tilde{\epsilon}_i\sqrt{\widehat{V(\mu(x_i))}}$ to the fitted values. This does not really work when the response is discrete-valued, however.

[[ATTN: Look up if anyone has a good trick for this]]

12.1.4 Modeling Dispersion

When we pick a family for the conditional distribution of Y, we get a predicted conditional variance function, $V(\mu(x))$. The actual conditional variance $\mathbb{V}[Y|X=x]$ may however not track this. When the variances are larger, the process is **over-dispersed**; when they are smaller, **under-dispersed**. Over-dispersion is more common and more worrisome. In many cases, it arises from some un-modeled aspect of the process — some unobserved heterogeneity, or some missed dependence. For instance, if we observe count data with an upper limit and use a binomial model, we're assuming that each "trial" within a data point is independent; positive correlation between the trials will give larger variance around the mean that the $mp(1-p)$ we'd expect[1].

The most satisfying solution to over-dispersion is to actually figure out where it comes from, and model its origin. Failing that, however, we can fall back on more "phenomenological" modeling. One strategy is to say that

$$\mathbb{V}[Y|X=x] = \phi(x)V(\mu(x)) \tag{12.10}$$

and try to estimate the function ϕ — a modification of the variance-estimation idea we saw in §10.3. In doing so, we need a separate estimate of $\mathbb{V}[Y|X=x_i]$. This can come from repeated measurements at the same value of x, or from the squared residuals at each data point. Once we have some noisy but independent estimate of $\mathbb{V}[Y|X=x_i]$, the ratio $\mathbb{V}[Y|X=x_i]/V(\mu(x_i))$ can be regressed on x_i to estimate ϕ. Some people recommend doing this step, itself, through a generalized linear or generalized additive model, with a gamma distribution for the response, so that the response is guaranteed to be positive.

12.1.5 Likelihood and Deviance

When dealing with GLMs, it is conventional to report not the log-likelihood, but the **deviance**. The deviance of a model with parameters (β_0, β) is defined as

$$D(\beta_0, \beta) = 2[\ell(\text{saturated}) - \ell(\beta_0, \beta)] \tag{12.11}$$

Here, $\ell(\beta_0, \beta)$ is the log-likelihood of our model, and $\ell(\text{saturated})$ is the log-likelihood of a **saturated** model which has one parameter per data point. Thus, models with high likelihoods will have low deviances, and vice versa. If our model is correct and has $p+1$ parameters in all (including the intercept), then the deviance will generally approach a χ^2 distribution asymptotically, with $n-(p+1)$ degrees of freedom; the factor of 2 in the definition is to ensure this.

For discrete response variables, the saturated model can usually ensure that $\Pr(Y=y_i|X=x_i) = 1$, so $\ell(\text{saturated}) = 0$, and deviance is just twice the negative log-likelihood. If there are multiple data points with the same value of x but different values of y, then $\ell(\text{saturated}) < 0$. In any case, even for repeated values of x or even continuous response variables, differences in deviance are

[1] If (for simplicity) all the trials have the same covariance ρ, then the variance of their sum is $mp(1-p) + m(m-1)\rho$ (why?).

just twice differences in log-likelihood: $D(\text{model}_1) - D(\text{model}_2) = 2[\ell(\text{model}_2) - \ell(\text{model}_1)]$.

12.1.5.1 Maximum Likelihood and the Choice of Link Function

Having chosen a family of conditional distributions, it may happen that when we write out the log-likelihood, the latter depends on the *both* the response variables y_i and the coefficients only through the product of y_i with some transformation of the conditional mean $\widehat{\mu}$:

$$\ell = \sum_{i=1}^{n} f(y_i, x_i) + y_i g(\widehat{\mu}_i) + h(\theta) \tag{12.12}$$

In the case of logistic regression, examining Eq. 11.8 (§11.2.1, p. 263) shows that the log-likelihood can be put in this form with $g(\widehat{\mu}_i) = \log \widehat{\mu}_i/(1-\widehat{\mu}_i)$. In the case of a Gaussian conditional distribution for Y, we would have $f = -y_i^2/2$, $g(\widehat{\mu}_i) = \widehat{\mu}_i$, and $h(\theta) = -\widehat{\mu}_i^2$. When the log-likelihood can be written in this form, $g(\cdot)$ is the "natural" transformation to apply to the conditional mean, i.e., the natural link function, and assures us that the solution to iterative least squares will converge on the maximum likelihood estimate.[2] Of course we are free to nonetheless use other transformations of the conditional expectation.

12.1.6 R: `glm`

As with logistic regression, the workhorse R function for all manner of GLMs is, simply, `glm`. The syntax is strongly parallel to that of `lm`, with the addition of a `family` argument that specifies the intended distribution of the response variable (`binomial`, `gaussian`, `poisson`, etc.), and, optionally, a link function appropriate to the family. (See `help(family)` for the details.) With `family="gaussian"` and an identity link function, its intended behavior is the same as `lm`.

[2] To be more technical, we say that a distribution with parameters θ is an **exponential family** if its probability density function at x is $\exp f(x) + T(x) \cdot g(\theta)/z(\theta)$, for some vector of statistics T and some transformation g of the parameters. (To ensure normalization, $z(\theta) = \int \exp(f(x) + T(x) \cdot g(\theta))dx$. Of course, if the sample space x is discrete, replace this integral with a sum.) We then say that $T(\cdot)$ are the "natural" or "canonical" **sufficient statistics**, and $g(\theta)$ are the "natural" parameters. Eq. 12.12 is picking out the natural parameters, presuming the response variable is itself the natural sufficient statistic. Many of the familiar families of distributions, like Gaussians, exponentials, gammas, Paretos, binomials and Poissons are exponential families. Exponential families are very important in classical statistical theory, and have deep connections to thermodynamics and statistical mechanics (where they're called "canonical ensembles", "Boltzmann distributions" or "Gibbs distributions" (Mandelbrot, 1962)), and to information theory (where they're "maximum entropy distributions", or "minimax codes" (Grünwald, 2007)). Despite their coolness, they are a rather peripheral topic for our sort of data analysis — though see Guttorp (1995) for examples of using them in modeling discrete processes. Any good book on statistical theory (e.g., Casella and Berger 2002) will have a fairly extensive discussion; Barndorff-Nielsen (1978) and Brown (1986) are comprehensive treatments.

12.2 Generalized Additive Models

In the development of generalized linear models, we use the link function g to relate the conditional mean $\widehat{\mu}(x)$ to the linear predictor $\eta(x)$. But really nothing in what we were doing required η to be *linear* in x. In particular, it all works perfectly well if η is an additive function of x. We form the effective responses z_i as before, and the weights w_i, but now instead of doing a linear regression on x_i we do an additive regression, using backfitting (or whatever). This gives us a generalized additive model (GAM).

Essentially everything we know about the relationship between linear models and additive models carries over. GAMs converge somewhat more slowly as n grows than do GLMs, but the former have less bias, and strictly include GLMs as special cases. The transformed (mean) response is related to the predictor variables not just through coefficients, but through whole partial response functions. If we want to test whether a GLM is well-specified, we can do so by comparing it to a GAM, and so forth.

In fact, one could even make $\eta(x)$ an arbitrary smooth function of x, to be estimated through (say) kernel smoothing of z_i on x_i. This is rarely done, however, partly because of curse-of-dimensionality issues, but also because, if one is going to go that far, one might as well just use kernels to estimate conditional distributions, as we will see in Chapter 14.

12.3 Further Reading

At our level of theory, good references on generalized linear and generalized additive models include Faraway (2006) and Wood (2006), both of which include extensive examples in R. Tutz (2012) offers an extensive treatment of GLMs with categorical response distributions, along with comparisons to other models for that task.

Overdispersion is the subject of a large literature of its own. All of the references just named discuss methods for it. Lambert and Roeder (1995) is worth mentioning for introducing some simple-to-calculate ways of detecting and describing over-dispersion which give some information about *why* the response is over-dispersed. One of these (the "relative variance curve") is closely related to the idea sketched above about estimating the dispersion factor.

Exercises

12.1 In binomial regression, we have $Y|X = x \sim \text{Binom}(n, p(x))$, where $p(x)$ follows a logistic model. Work out the link function $g(\mu)$, the variance function $V(\mu)$, and the weights w, assuming that n is known and not random.

12.2 Problem set 12, on predicting the death rate in Chicago, is a good candidate for using Poisson regression. Repeat the exercises in that problem set with Poisson-response GAMs. How do the estimated functions change? Why is this any different from just taking the log of the death counts, as suggested in that problem set?

13

Classification and Regression Trees

[[TODO: Notes taken from another course; integrate]]

So far, the models we've worked with have been built on the principle of every point in the data set contributing (at least potentially) to every prediction. An alternative is to divide up, or **partition**, the data set, so that each prediction will only use points from one chunk of the space. If this partition is done in a recursive or hierarchical manner, we get a **prediction tree**, which comes in two varieties, **regression trees** and **classification trees**. These may seem too crude to actually work, but they can, in fact, be both powerful and computationally efficient.

13.1 Prediction Trees

The basic idea is simple. We want to predict some variable Y from other variables $X_1, X_2, \ldots X_p$. We do this by growing a binary tree. At each internal node in the tree, we apply a test to one of the predictor variables, say X_i. Depending on the outcome of the test, we go to either the left or the right sub-branch of the tree. Eventually we come to a leaf node, where we make a prediction. This prediction aggregates or averages all the training data points which reach that leaf. Figure 13.1 should help clarify this.

Why do this? Predictors like linear or polynomial regression are **global models**, where a single predictive formula is supposed to hold over the entire data space. When the data has lots of variables which interact in complicated, nonlinear ways, assembling a single global model can be very difficult, and hopelessly confusing when you do succeed. As we've seen, non-parametric smoothers try to fit models **locally** and then paste them together, but again they can be hard to interpret. (Additive models are at least pretty easy to grasp.)

An alternative approach to nonlinear prediction is to sub-divide, or **partition**, the space into smaller regions, where the relationships between variables are more manageable. We then partition the sub-divisions again — this is **recursive partitioning** (or **hierarchical** partitioning) — until finally we get to chunks of the space which are so tame that we can fit simple models to them. The global model thus has two parts: one is just the recursive partition, the other is a simple model for each cell of the partition.

Now look back at Figure 13.1 and the description which came before it. Prediction trees use the tree to represent the recursive partition. Each of the **terminal nodes**, or **leaves**, of the tree represents a cell of the partition, and has attached

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

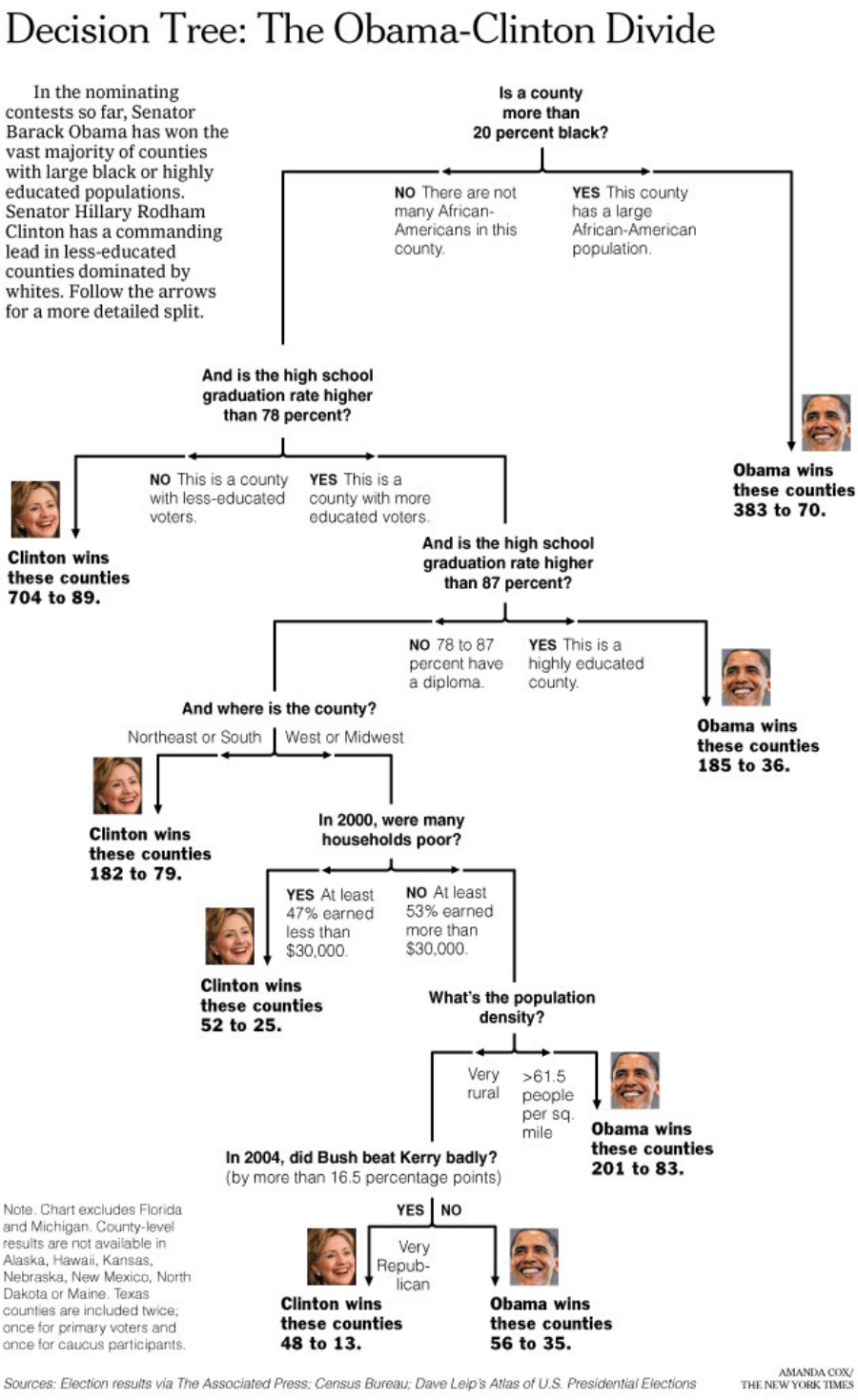

Decision Tree: The Obama-Clinton Divide
In the nominating contests so far, Senator Barack Obama has won the vast majority of counties with large black or highly educated populations. Senator Hillary Rodham Clinton has a commanding lead in less-educated counties dominated by whites. Follow the arrows for a more detailed split.
Is a county more than 20 percent black?
NO There are not many African-Americans in this county.
YES This county has a large African-American population.
Obama wins these counties 383 to 70.
And is the high school graduation rate higher than 78 percent?
NO This is a county with less-educated voters.
YES This is a county with more educated voters.
Clinton wins these counties 704 to 89.
And is the high school graduation rate higher than 87 percent?
NO 78 to 87 percent have a diploma.
YES This is a highly educated county.
Obama wins these counties 185 to 36.
And where is the county?
Northeast or South
West or Midwest
Clinton wins these counties 182 to 79.
In 2000, were many households poor?
YES At least 47% earned less than $30,000.
NO At least 53% earned more than $30,000.
Clinton wins these counties 52 to 25.
What's the population density?
Very rural
>61.5 people per sq. mile
Obama wins these counties 201 to 83.
In 2004, did Bush beat Kerry badly?
(by more than 16.5 percentage points)
YES
NO
Very Republican
Clinton wins these counties 48 to 13.
Obama wins these counties 56 to 35.
Note. Chart excludes Florida and Michigan. County-level results are not available in Alaska, Hawaii, Kansas, Nebraska, New Mexico, North Dakota or Maine. Texas counties are included twice; once for primary voters and once for caucus participants.
Sources: Election results via The Associated Press; Census Bureau; Dave Leip's Atlas of U.S. Presidential Elections
AMANDA COX/ THE NEW YORK TIMES

Figure 13.1 Classification tree for county-level outcomes in the 2008 Democratic Party primary (as of April 16), by Amanada Cox for the New York *Times*. [[TODO: Get figure permission!]]

to it a simple model which applies in that cell only. A point x **belongs** to a leaf if x falls in the corresponding cell of the partition. To figure out which cell we are in, we start at the **root node** of the tree, and ask a sequence of questions about the predictor variables. The **interior nodes** are labeled with questions, and the edges or branches between them labeled by the answers. Which question we ask next depends on the answers to previous questions. In the classic version, each question refers to only a single attribute, and has a yes or no answer, e.g., "Is `HSGrad` > 0.78?" or "Is `Region` == MIDWEST?" The variables can be of any combination of types (continuous, discrete but ordered, categorical, etc.). You could do more-than-binary questions, but that can always be accommodated as a larger binary tree. Asking questions about multiple variables at once is, again, equivalent to asking multiple questions about single variables.

That's the recursive partition part; what about the simple local models? For classic regression trees, the model in each cell is just a *constant* estimate of Y. That is, suppose the points $(x_i, y_i), (x_2, y_2), \ldots (x_c, y_c)$ are all the samples belonging to the leaf-node l. Then our model for l is just $\hat{y} = \frac{1}{c}\sum_{i=1}^{c} y_i$, the sample mean of the response variable in that cell. This is a piecewise-constant model.[1] There are several advantages to this:

- Making predictions is fast (no complicated calculations, just looking up constants in the tree).
- It's easy to understand what variables are important in making the prediction (look at the tree).
- If some variables are missing, we might not be able to go all the way down the tree to a leaf, but we can still make a prediction by averaging all the leaves in the sub-tree we do reach.
- The model gives a jagged response, so it can work when the true regression surface is not smooth. If it is smooth, though, the piecewise-constant surface can approximate it arbitrarily closely (with enough leaves).
- There are fast, reliable algorithms to learn these trees.

A last analogy before we go into some of the mechanics. One of the most comprehensible non-parametric methods is k-nearest-neighbors: find the points which are most similar to you, and do what, on average, they do. There are two big drawbacks to it: first, you're defining "similar" entirely in terms of the inputs, not the response; second, k is constant everywhere, when some points just might have more very-similar neighbors than others. Trees get around both problems: leaves correspond to regions of the input space (a neighborhood), but one where the *responses* are similar, as well as the inputs being nearby; and their size can vary arbitrarily. Prediction trees are, in a way, *adaptive* nearest-neighbor methods.

[1] We could instead fit, say, a different linear regression for the response in each leaf node, using only the data points in that leaf (and using dummy variables for non-quantitative variables). This would give a piecewise-linear model, rather than a piecewise-constant one. If we've built the tree well, however, all the points in each leaf are pretty similar, so the regression surface would be nearly constant anyway.

13.2 Regression Trees

Let's start with an example.

13.2.1 Example: California Real Estate Again

We'll revisit the Califonia house-price data from Chapter 8, where we try to predict the median house price in each census tract of California from the attributes of the houses and of the inhabitants of the tract. We'll try growing a regression tree for this.

There are several R packages for regression trees; the easiest one is called, simply, `tree` (Ripley, 2015).

```
calif <- read.table("http://www.stat.cmu.edu/~cshalizi/350/hw/06/cadata.dat", header = TRUE)
require(tree)
treefit <- tree(log(MedianHouseValue) ~ Longitude + Latitude, data = calif)
```

This does a tree regression of the log price on longitude and latitude. What does this look like? Figure 13.2 shows the tree itself; Figure 13.3 shows the partition, overlaid on the actual prices in the state. (The ability to show the partition is why I picked only two input variables.)

Qualitatively, this looks like it does a fair job of capturing the interaction between longitude and latitude, and the way prices are higher around the coasts and the big cities. Quantitatively, the error isn't bad:

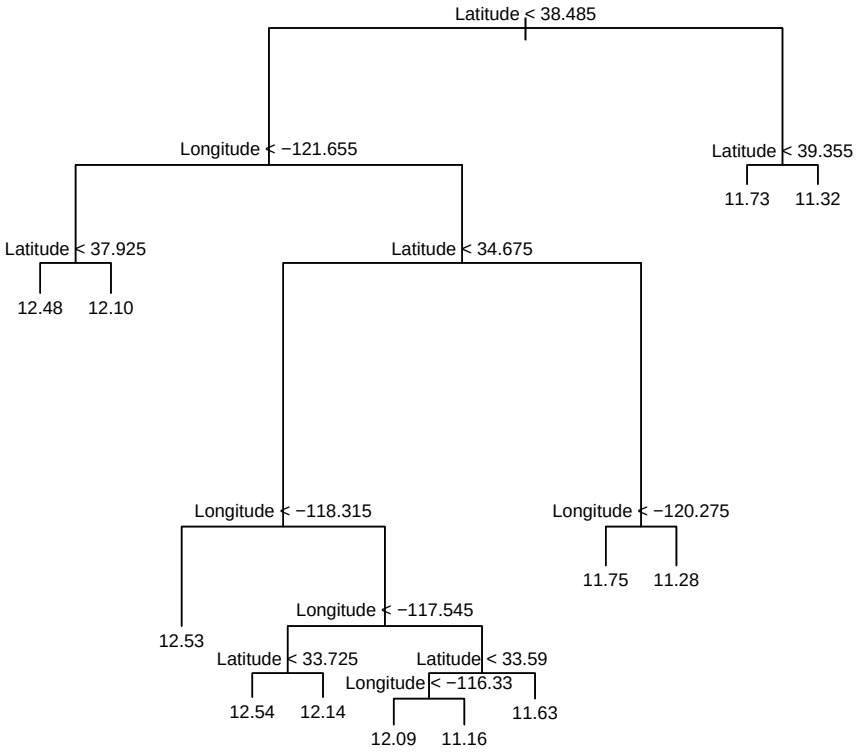
```
summary(treefit)
##
## Regression tree:
## tree(formula = log(MedianHouseValue) ~ Longitude + Latitude,
##     data = calif)
## Number of terminal nodes:  12
## Residual mean deviance:  0.1662 = 3429 / 20630
## Distribution of residuals:
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## -2.75900 -0.26080 -0.01359  0.00000  0.26310  1.84100
```

Here "deviance" is just mean squared error; this gives us an RMS error of 0.41, which is higher than the smooth non-linear models in Chapter 8, but not shocking since we're using only two variables, and have only twelve leaves.

The flexibility of a tree is basically controlled by how many leaves they have, since that's how many cells they partition things into. The `tree` fitting function has a number of controls settings which limit how much it will grow — each node has to contain a certain number of points, and adding a node has to reduce the error by at least a certain amount. The default for the latter, `mindev`, is 0.01; let's turn it down and see what happens.

```
treefit2 <- tree(log(MedianHouseValue) ~ Longitude + Latitude, data = calif, mindev = 0.001)
```

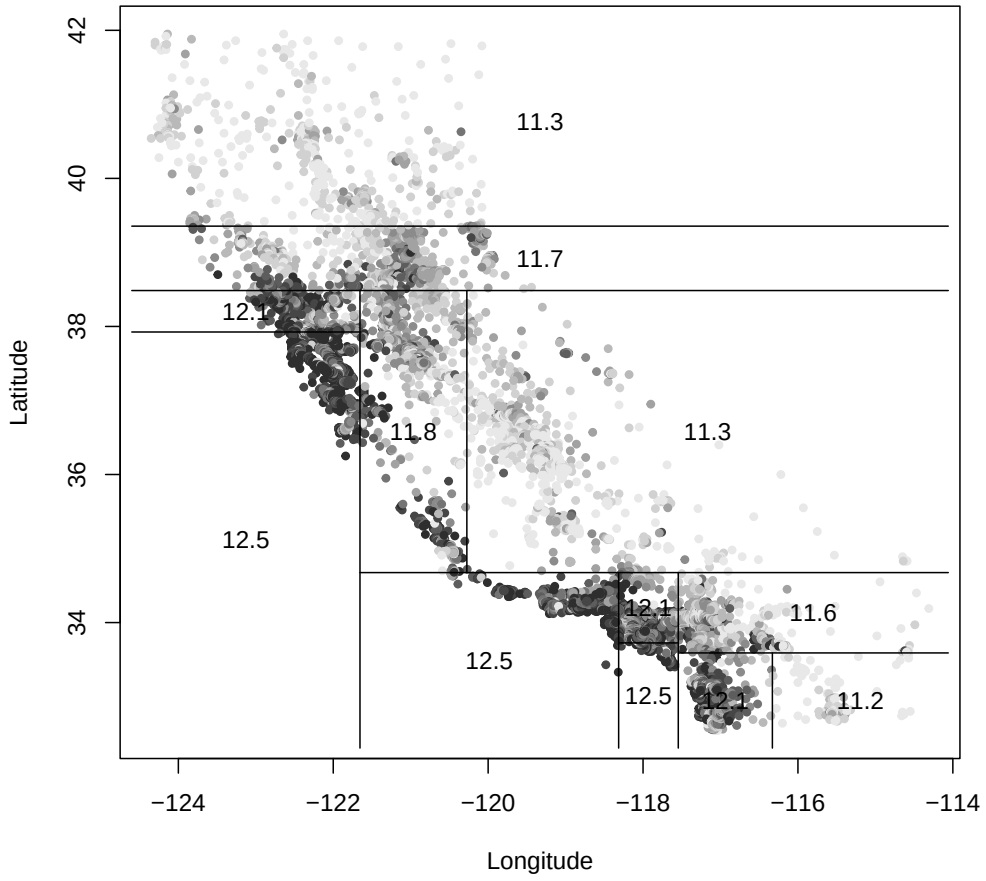
Figure 13.4 shows the tree itself; with 68 nodes, the plot is fairly hard to read, but by zooming in on any part of it, you can check what it's doing. Figure 13.5 shows the corresponding partition. It's obviously much finer-grained than that




```
plot(treefit)
text(treefit, cex = 0.75)
```

Figure 13.2 Regression tree for predicting California housing prices from geographic coordinates. At each internal node, we ask the associated question, and go to the left child if the answer is "yes", to the right child if the answer is "no". Note that leaves are labeled with *log* prices; the plotting function isn't flexible enough, unfortunately, to apply transformations to the labels.

in Figure 13.3, and does a better job of matching the actual prices (RMS error 0.32). More interestingly, it doesn't just uniformly divide up the big cells from the first partition; some of the new cells are very small, others quite large. The metropolitan areas get a lot more detail than the Mojave.




```
price.deciles <- quantile(calif$MedianHouseValue, 0:10/10)
cut.prices <- cut(calif$MedianHouseValue, price.deciles, include.lowest = TRUE)
plot(calif$Longitude, calif$Latitude, col = grey(10:2/11)[cut.prices], pch = 20,
    xlab = "Longitude", ylab = "Latitude")
partition.tree(treefit, ordvars = c("Longitude", "Latitude"), add = TRUE)
```

Figure 13.3 Map of actual median house prices (color-coded by decile, darker being more expensive), and the partition of the `treefit` tree.

Of course there's nothing magic about the geographic coordinates, except that they make for pretty plots. We can include all the predictor variables in our model

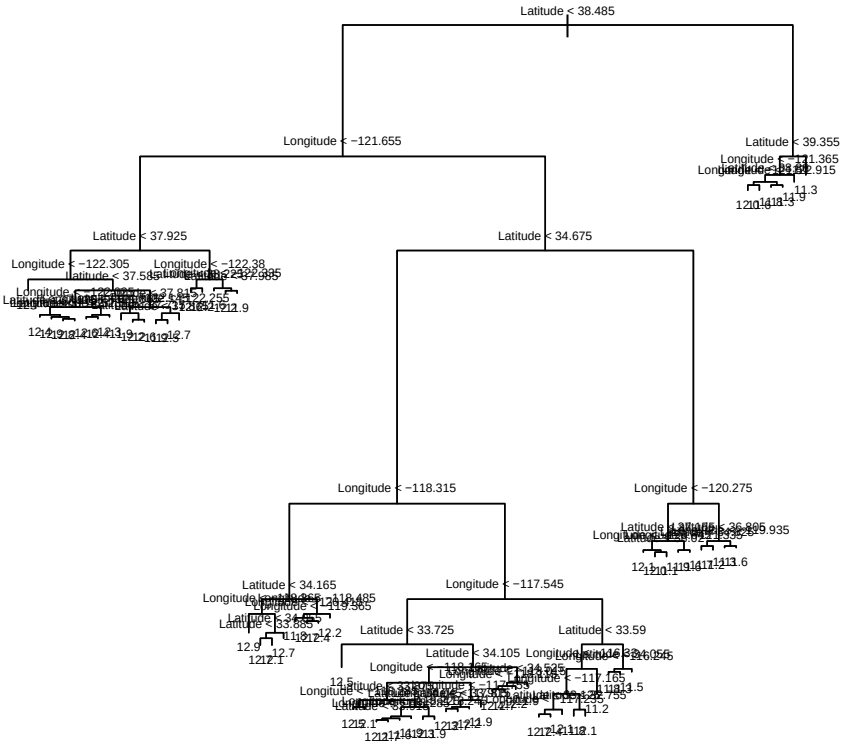
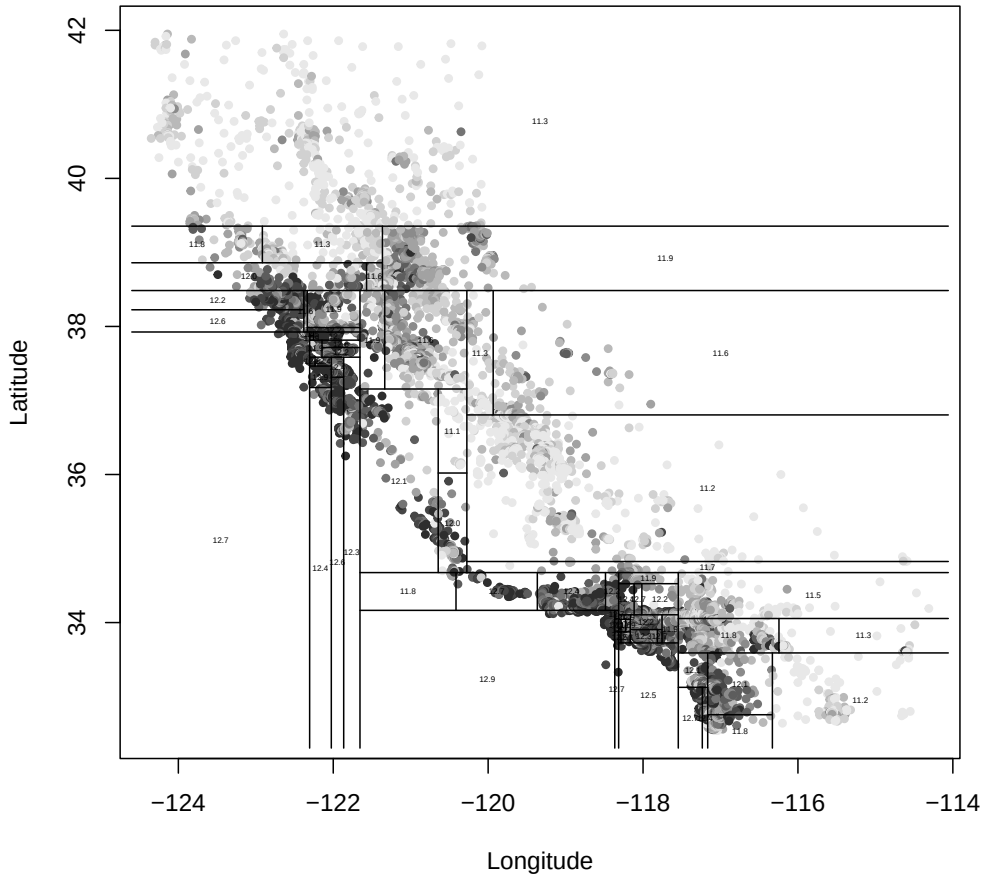


Figure 13.4 As Figure 13.2, but allowing splits for smaller reductions in error (`mindev=0.001` rather than the default `mindev=0.01`). Then fact that the plot is nearly illegible is deliberate.

```
treefit3 <- tree(log(MedianHouseValue) ~ ., data = calif)
```

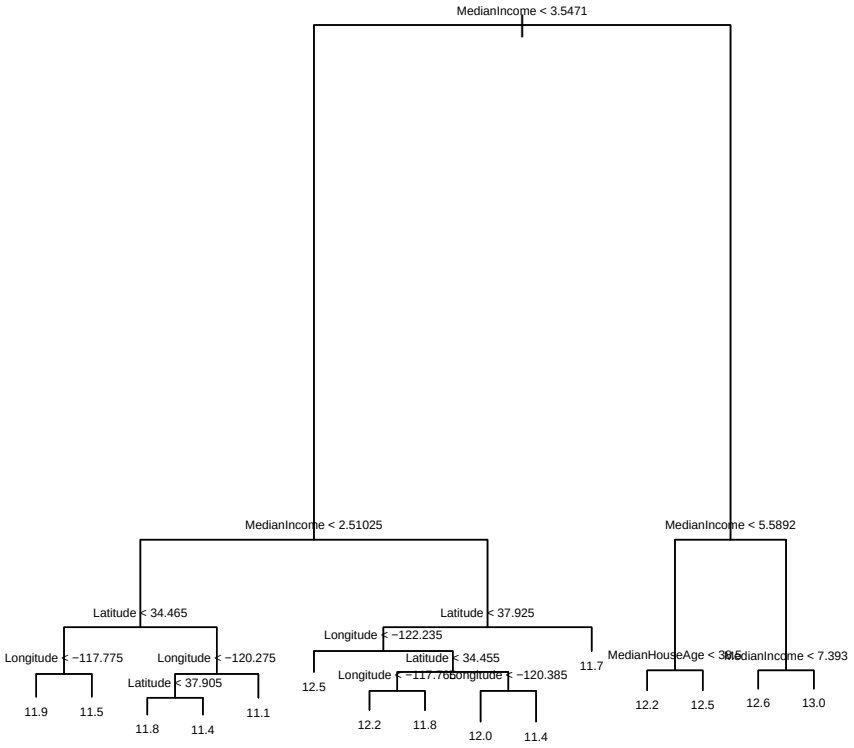
with the result shown in Figure 13.6. This model has fifteen leaves, as opposed to sixty-eight for `treefit2`, but the RMS error is almost as good (0.36). This is highly interactive: latitude and longitude are only used if the income level is sufficiently low. (Unfortunately, this does mean that we don't have a spatial partition to compare to the previous ones, but we *can* map the predictions; Figure 13.7.) Many of the variables, while they were available to the tree fit, aren't used at all.




```
plot(calif$Longitude, calif$Latitude, col = grey(10:2/11)[cut.prices], pch = 20,
    xlab = "Longitude", ylab = "Latitude")
partition.tree(treefit2, ordvars = c("Longitude", "Latitude"), add = TRUE, cex = 0.3)
```

Figure 13.5 Partition for `treefit2`. Note the high level of detail around the cities, as compared to the much coarser cells covering rural areas where variations in prices are less extreme.

Now let's turn to how we actually grow these trees.



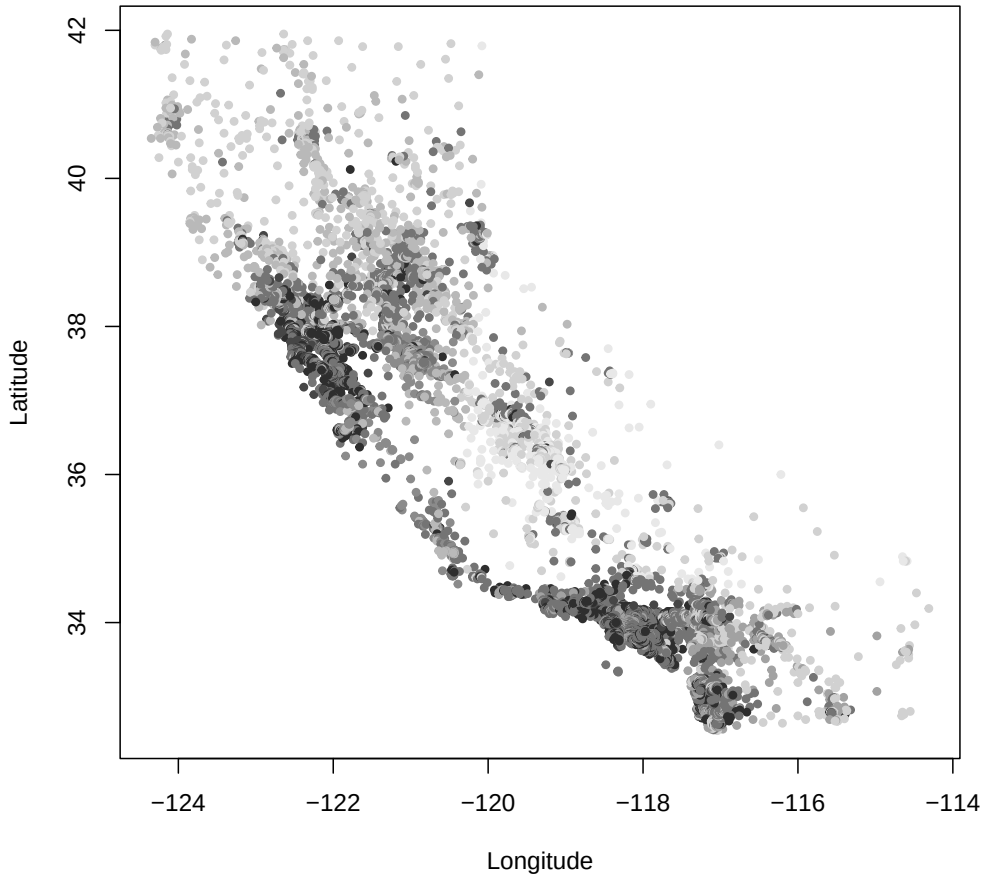

```
plot(treefit3)
text(treefit3, cex = 0.5, digits = 3)
```

Figure 13.6 Regression tree for log price when all other variables are included as (potential) predictors. Note that the tree ignores many variables.

13.2.2 Regression Tree Fitting

Once we fix the tree, the local models are completely determined, and easy to find (we just average), so all the effort should go into finding a good tree, which is to say into finding a good partitioning of the data.

Ideally, we maximize the information the partition gives us about the response variable. Since we are doing regression, what we would really like is for the condi-




```
cut.predictions <- cut(predict(treefit3), log(price.deciles), include.lowest = TRUE)
plot(calif$Longitude, calif$Latitude, col = grey(10:2/11)[cut.predictions], pch = 20,
    xlab = "Longitude", ylab = "Latitude")
```

Figure 13.7 Predicted prices for the `treefit3` model. Same color scale as in previous plots (where dots indicated actual prices).

tional mean $\mathbb{E}[Y|X = x]$ to be nearly constant in x over each cell of the partition, and for adjoining cells to have distinct expected values. (It's OK if two cells of the partition far apart have similar average values.) It's too hard to do this directly, so we do a greedy search. We start by finding the one binary question we can ask about the predictors which maximizes the information we get about the average

value of Y; this gives us our root node and two daughter nodes.[2] At each daughter node, we repeat our initial procedure, asking which question would give us the maximum information about the average value of Y, given where we already are in the tree. We repeat this recursively.

Every recursive algorithm needs to know when it's done, a **stopping criterion**. Here this means when to stop trying to split nodes. Obviously nodes which contain only one data point cannot be split, but giving each observations its own leaf is unlikely to generalize well. A more typical criterion is something like: halt when each child would contain less than five data points, or when splitting increases the information by less than some threshold. Picking the criterion is important to get a good tree, so we'll come back to it presently.

To really make this work, we need to be precise about "information about the average value of Y". This can be measured straightforwardly by the mean squared error. The MSE for a tree T is

$$MSE(T) = \frac{1}{n} \sum_{c \in \text{leaves}(T)} \sum_{i \in c} (y_i - m_c)^2 \tag{13.1}$$

where $m_c = \frac{1}{n_c} \sum_{i \in c} y_i$, the prediction for leaf c.

The basic regression-tree-growing algorithm then is as follows:

1. Start with a single node containing all points. Calculate m_c and MSE.
2. If all the points in the node have the same value for all the input variables, stop. Otherwise, search over all binary splits of all variables for the one which will reduce MSE as much as possible. If the largest decrease in MSE would be less than some threshold δ, or one of the resulting nodes would contain less than q points, stop. Otherwise, take that split, creating two new nodes.
3. In each new node, go back to step 1.

Trees use only one variable at each step. If multiple variables are equally good, which one is chosen is a matter of chance, or arbitrary programming decisions.

One problem with the straight-forward algorithm I've just given is that it can stop too early, in the following sense. There can be variables which are not very informative themselves, but which lead to very informative subsequent splits. This suggests a problem with stopping when the decrease in S becomes less than some δ. Similar problems can arise from arbitrarily setting a minimum number of points q per node.

A more successful approach to finding regression trees uses the idea of cross-validation (Chapter 3), especially k-fold cross-validation. We initially grow a large tree, looking only at the error on the training data. (We might even set $q = 1$ and $\delta = 0$ to get the largest tree we can.) This tree is generally too large and will over-fit the data.

The issue is basically about the number of leaves in the tree. For a given number of leaves, there is a unique best tree. As we add more leaves, we can only lower the bias, but we also increase the variance, since we have to estimate more. At

[2] Mixing botanical and genealogical metaphors for trees is ugly, but I can't find a way around it.

any finite sample size, then, there is a tree with a certain number of leaves which will generalize better than any other. We would like to find this optimal number of leaves.

The reason we start with a big (lush? exuberant? spreading?) tree is to make sure that we've got an upper bound on the optimal number of leaves. Thereafter, we consider simpler trees, which we obtain by **pruning** the large tree. At each pair of leaves with a common parent, we evaluate the error of the tree on the testing data, and also of the sub-tree, which removes those two leaves and puts a leaf at the common parent. We then prune that branch of the tree, and so forth until we come back to the root. Starting the pruning from different leaves may give multiple pruned trees with the same number of leaves; we'll look at which sub-tree does best on the testing set. The reason this is superior to arbitrary stopping criteria, or to rewarding parsimony as such, is that it directly checks whether the extra capacity (nodes in the tree) pays for itself by improving generalization error. If it does, great; if not, get rid of the complexity.

There are lots of other cross-validation tricks for trees. One cute one is to alternate growing and pruning. We divide the data into two parts, as before, and first grow and then prune the tree. We then exchange the role of the training and testing sets, and try to grow our pruned tree to fit the second half. We then prune again, on the first half. We keep alternating in this manner until the size of the tree doesn't change.

13.2.2.1 Cross-Validation and Pruning in R

The `tree` package contains functions `prune.tree` and `cv.tree` for pruning trees by cross-validation.

The function `prune.tree` takes a tree you fit by `tree`, and evaluates the error of the tree and various prunings of the tree, all the way down to the stump. The evaluation can be done either on new data, if supplied, or on the training data (the default). If you ask it for a particular size of tree, it gives you the best pruning of that size[3]. If you don't ask it for the best tree, it gives an object which shows the number of leaves in the pruned trees, and the error of each one. This object can be plotted.

```
my.tree <- tree(y ~ x1 + x2, data = my.data)
prune.tree(my.tree, best = 5)
prune.tree(my.tree, best = 5, newdata = test.set)
my.tree.seq <- prune.tree(my.tree)
plot(my.tree.seq)
my.tree.seq$dev
opt.trees <- which(my.tree.seq$dev == min(my.tree.seq$dev))
min(my.tree.seq$size[opt.trees])
```

Finally, `prune.tree` has an optional `method` argument. The default is `method="deviance"`,

[3] Or, if there is no tree with that many leaves, the smallest number of leaves $\geq$ the requested size.

which fits by minimizing the mean squared error (for continuous responses) or the negative log likelihood (for discrete responses; see below).[4]

The function `cv.tree` does k-fold cross-validation (default is $k = 10$). It requires as an argument a fitted tree, and a function which will take that tree and new data. By default, this function is `prune.tree`.

```
my.tree.cv <- cv.tree(my.tree)
```

The type of output of `cv.tree` is the same as the function it's called on. If I do

```
cv.tree(my.tree, best = 19)
```

I get the best tree (per cross-validation) of no more than 19 leaves. If I do

```
cv.tree(my.tree)
```

I get information about the cross-validated performance of the whole *sequence* of pruned trees, e.g., `plot(cv.tree(my.tree))`. Optional arguments to `cv.tree` can include the number of folds, and any additional arguments for the function it applies (e.g., any arguments taken by `prune`).

To illustrate, think back to `treefit2`, which predicted predicted California house prices based on geographic coordinates, but had a very large number of nodes because the tree-growing algorithm was told to split at the least provcation. Figure 13.8 shows the size/performance trade-off. Figures 13.9 and 13.10 show the result of pruning to the smallest size compatible with minimum cross-validated error.

13.2.3 Uncertainty in Regression Trees

Even when we are making point predictions, we have some uncertainty, because we've only seen a finite amount of data, and this is not an *entirely* representative sample of the underlying probability distribution. With a regression tree, we can separate the uncertainty in our predictions into two parts. First, we have some uncertainty in what our predictions should be, *assuming* the tree is correct. Second, we may of course be wrong about the tree.

The first source of uncertainty — imprecise estimates of the conditional means within a given partition — is fairly easily dealt with. We can consistently estimate the standard error of the mean for leaf c just like we would for any other mean of IID samples. The second source is more troublesome; as the response values shift, the tree itself changes, and discontinuously so, tree shape being a discrete variable. What we want is some estimate of how different the tree could have been, had we just drawn a different sample from the same source distribution.

One way to estimate this, from the data at hand, is to use bootstrapping (ch.

[4] With discrete responses, you may get better results by saying `method="misclass"`, which looks at the misclassification rate.

6). It is important that we apply the bootstrap to the predicted values, which can change smoothly if we make a tiny perturbation to the distribution, and not to the shape of the tree itself (which can only change abruptly).

13.3 Classification Trees

Classification trees work just like regression trees, only they try to predict a discrete category (the class), rather than a numerical value. The variables which go into the classification — the inputs — can be numerical or categorical themselves, the same way they can with a regression tree. They are useful for the same reasons regression trees are — they provide fairly comprehensible predictors in situations where there are many variables which interact in complicated, nonlinear ways.

We find classification trees in almost the same way we found regression trees: we start with a single node, and then look for the binary distinction which gives us the most information about the class. We then take each of the resulting new nodes and repeat the process there, continuing the recursion until we reach some stopping criterion. The resulting tree will often be too large (i.e., over-fit), so we prune it back using (say) cross-validation. The differences from regression-tree growing have to do with (1) how we measure information, (2) what kind of predictions the tree makes, and (3) how we measure predictive error.

13.3.1 Measuring Information

The response variable Y is categorical, so we can use information theory to measure how much we learn about it from knowing the value of another discrete variable A:

$$I[Y; A] \equiv \sum_a \Pr(A = a)\, I[Y; A = a] \tag{13.2}$$

where

$$I[Y; A = a] \equiv H[Y] - H[Y|A = a] \tag{13.3}$$

and you remember the definitions of entropy $H[Y]$ and conditional entropy $H[Y|A = a]$,

$$H[Y] \equiv \sum_y -\Pr(Y = y) \log_2 \Pr(Y = y) \tag{13.4}$$

and

$$H[Y|A = a] \equiv \sum_y -\Pr(Y = y|A = a) \log_2 \Pr(Y = y|A = a) \tag{13.5}$$

$I[Y; A = a]$ is how much our uncertainty about Y decreases from knowing that $A = a$. (Less subjectively: how much less variable Y becomes when we go from the full population to the sub-population where $A = a$.) $I[Y; A]$ is how much our uncertainty about Y shrinks, on average, from knowing the value of A.

For classification trees, A isn't (necessarily) one of the predictors, but rather

the answer to some question, generally binary, about one of the predictors X, i.e., $A = \mathbf{1}_{\mathcal{A}}(X)$ for some set $\mathcal{A}$. This doesn't change any of the math above, however. So we chose the question in the first, root node of the tree so as to maximize $I[Y; A]$, which we calculate from the formula above, using the relative frequencies in our data to get the probabilities.

When we want to get good questions at subsequent nodes, we have to take into account what we know already at each stage. Computationally, we do this by computing the probabilities and informations using only the cases in that node, rather than the complete data set. (Remember that we're doing *recursive* partitioning, so at each stage the sub-problem looks just like a smaller version of the original problem.) Mathematically, what this means is that if we reach the node when $A = a$ and $B = b$, we look for the question C which maximizes $I[Y; C|A = a, B = b]$, the information *conditional* on $A = a$, $B = b$. Algebraically,

$$I[Y; C|A = a, B = b] = H[Y|A = a, B = b] - H[Y|A = a, B = b, C] \quad (13.6)$$

Computationally, rather than looking at all the cases in our data set, we just look at the ones where $A = a$ and $B = b$, and calculate as though that were all the data. Also, notice that the first term on the right-hand side, $H[Y|A = a, B = b]$, does not depend on the next question C. So rather than maximizing $I[Y; C|A = a, B = b]$, we can just minimize $H[Y|A = a, B = b, C]$.

13.3.2 Making Predictions

There are two kinds of predictions which a classification tree can make. One is a **point** prediction, a single guess as to the class or category: to say "this is a flower" or "this is a tiger" and nothing more. The other, a **distributional prediction**, gives a *probability* for each class. This is slightly more general, because if we need to extract a point prediction from a probability forecast we can always do so, but we can't go in the other direction.

For probability forecasts, each terminal node in the tree gives us a distribution over the classes. If the terminal node corresponds to the sequence of answers $A = a, B = b, \ldots Q = q$, then ideally this would give us $\Pr(Y = y|A = a, B = b, \ldots Q = q)$ for each possible value y of the response. A simple way to get close to this is to use the empirical relative frequencies of the classes in that node. E.g., if there are 33 cases at a certain leaf, 22 of which are tigers and 11 of which are flowers, the leaf should predict "tiger with probability 2/3, flower with probability 1/3". This is the **maximum likelihood** estimate of the true probability distribution, and we'll write it $\widehat{\Pr}(\cdot)$.

Incidentally, while the empirical relative frequencies are consistent estimates of the true probabilities under many circumstances, nothing particularly *compells* us to use them. When the number of classes is large relative to the sample size, we may easily fail to see any samples at all of a particular class. The empirical relative frequency of that class is then zero. This is good if the actual probability is zero, not so good otherwise. (In fact, under the negative log-likelihood error discussed below, it's infinitely bad, because we will eventually see that class, but

our model will say it's impossible.) The empirical relative frequency estimator is in a sense too reckless in following the data, without allowing for the possibility that it the data are wrong; it may under-smooth. Other probability estimators "shrink away" or "back off" from the empirical relative frequencies; Exercise 1 involves one such estimator.

For point forecasts, the best strategy depends on the loss function. If it is just the mis-classification rate, then the best prediction at each leaf is the class with the highest conditional probability in that leaf. With other loss functions, we should make the guess which minimizes the expected loss. But this leads us to the topic of measuring error.

13.3.3 Measuring Error

There are three common ways of measuring error for classification trees, or indeed other classification algorithms: misclassification rate, expected loss, and normalized negative log-likelihood, a.k.a. **cross-entropy**.

13.3.3.1 Misclassification Rate

We've already seen this: it's the fraction of cases assigned to the wrong class.

13.3.3.2 Average Loss

The idea of the average loss is that some errors are more costly than others. For example, we might try classifying cells into "cancerous" or "not cancerous" based on their gene expression profiles. If we think a healthy cell from someone's biopsy is cancerous, we refer them for further tests, which are frightening and unpleasant, but not, as the saying goes, the end of the world. If we think a cancer cell is healthy, th consequences are much more serious! There will be a different cost for each combination of the real class and the guessed class; write L_{ij} for the cost ("loss") we incur by saying that the class is j when it's really i.

For an observation x, the classifier gives class probabilities $\Pr(Y = i|X = x)$. Then the expected cost of predicting j is:

$$\text{Loss}(Y = j|X = x) = \sum_i L_{ij} \Pr(Y = i|X = x) \tag{13.7}$$

A cost matrix might look as follows

	prediction	
truth	"cancer"	"healthy"
"cancer"	0	100
"healthy"	1	0

We run an observation through the tree and wind up with class probabilities $(0.4, 0.6)$. The most likely class is "healthy", but it is not the most cost-effective decision. The expected cost of predicting "cancer" is $0.4 * 0 + 0.6 * 1 = 0.6$, while the expected cost of predicting "healthy" is $0.4 * 100 + 0.6 * 0 = 40$. The probability

of Y = "healthy" must be 100 times higher than that of Y = "cancer" before "cancer" is a cost-effective prediction.

Notice that if our estimate of the class probabilities is very bad, we can go through the math above correctly, but still come out with the wrong answer. If our estimates were exact, however, we'd always be doing as well as we could, given the data.

You can show (Exercise 13.6) that if the costs are symmetric, we get the misclassification rate back as our error function, and should always predict the most likely class.

13.3.3.3 Likelihood and Cross-Entropy

The normalized negative log-likelihood is a way of looking not just at whether the model made the wrong call, but whether it made the wrong call with confidence or tentatively. ("Often wrong, never in doubt" is *not* a good way to go through life.) More precisely, this loss function for a model Q is

$$L(\text{data}, Q) = -\frac{1}{n}\sum_{i=1}^{n} \log Q(Y = y_i | X = x_i) \tag{13.8}$$

where $Q(Y = y|X = x)$ is the conditional probability the model predicts. If perfect classification were possible, i.e., if Y were a function of X, then the best classifier would give the actual value of Y a probability of 1, and $L = 0$. If there is some irreducible uncertainty in the classification, then the best possible classifier would give $L = H[Y|X]$, the conditional entropy of Y given the inputs X. Less-than-ideal predictors have $L > H[Y|X]$. To see this, try re-write L so we sum over values rather than data-points:

$$\begin{aligned}
L &= -\frac{1}{n}\sum_{x,y} N(Y = y, X = x) \log Q(Y = y|X = x) \\
&= -\sum_{x,y} \widehat{\Pr}(Y = y, X = x) \log Q(Y = y|X = x) \\
&= -\sum_{x,y} \widehat{\Pr}(X = x) \widehat{\Pr}(Y = y|X = x) \log Q(Y = y|X = x) \\
&= -\sum_{x} \widehat{\Pr}(X = x) \sum_{y} \widehat{\Pr}(Y = y|X = x) \log Q(Y = y|X = x)
\end{aligned}$$

If the quantity in the log was $\Pr(Y = y|X = x)$, this would be $H[Y|X]$. Since it's the model's estimated probability, rather than the real probability, it turns out that this is always *larger* than the conditional entropy. L is also called the **cross-entropy** for this reason.

There is a slightly subtle issue here about the difference between the in-sample loss, and the expected generalization error or risk. $N(Y = y, X = x)/n = \widehat{\Pr}(Y = y, X = x)$, the empirical relative frequency or empirical probability. The law of large numbers says that this converges to the true probability, $N(Y = y, X = x)/n \to \Pr(Y = y, X = x)$ as $n \to \infty$. Consequently, the model which

minimizes the cross-entropy in sample may not be the one which minimizes it on future data, though the two ought to converge. Generally, the in-sample cross-entropy is lower than its expected value.

Notice that to compare two models, or the same model on two different data sets, etc., we do not need to know the true conditional entropy $H[Y|X]$. All we need to know is that L is smaller the closer we get to the true class probabilities. If we could get L down to the cross-entropy, we would be exactly reproducing all the class probabilities, and then we could use our model to minimize any loss function we liked (as we saw above).[5]

13.3.3.4 Neyman-Pearson Approach

Using a loss function which assigns different weights to different error types has two noticeable drawbacks. First of all, we have to *pick* the weights, and this is often quite hard to do. Second, whether our classifier will do well in the future depends on getting the same *proportion* of cases in the future. Suppose that we're developing a tree to classify cells as cancerous or not from their gene expression profiles. We will probably want to include lots of cancer cells in our training data, so that we can get a good idea of what cancers look like, biochemically. But, fortunately, most cells are *not* cancerous, so if doctors start applying our test to their patients, they're going to find that it massively over-diagnoses cancer — it's been calibrated to a sample where the proportion (cancer):(healthy) is, say, 1:1, rather than, say, 1:20.[6]

There is an alternative to weighting which deals with both of these issues, and deserves to be better known and more widely-used than it is. This was introduced by Scott and Nowak (2005), under the name of the "Neyman-Pearson approach" to statistical learning. The reasoning goes as follows.

When we do a binary classification problem, we're really doing a hypothesis test, and the central issue in hypothesis testing, as first recognized by Neyman and Pearson, is to distinguish between the rates of different *kinds* of errors: false positives and false negatives, false alarms and misses, type I and type II. The Neyman-Pearson approach to designing a hypothesis test is to first fix a limit on the false positive probability, the **size** of the test, canonically α. Then, among all tests of size α, we want to minimize the false negative rate, or equivalently maximize the power, β.

In the traditional theory of testing, we know the distribution of the data under the null and alternative hypotheses, and so can (in principle) calculate α and β for any given test. This is *not* the case in many applied problems, but then we

[5] Technically, if our model gets the class probabilities right, then the model's predictions are just as informative as the original data. We then say that the predictions are a **sufficient statistic** for forecasting the class. In fact, if the model gets the exact probabilities wrong, but has the correct partition of the variable space, then its prediction is still a sufficient statistic. Under any loss function, the optimal strategy can be implemented using *only* a sufficient statistic, rather than needing the full, original data. This is an interesting but much more advanced topic; see, e.g., Blackwell and Girshick (1954) for details.

[6] Cancer is rarer than that, but realistically doctors aren't going to run a test like this unless they have some reason to suspect cancer might be present.

often do have large samples generated under both distributions (depending on the class of the data point). If we fix α, we can ask, for any classifier — say, a tree — whether its false alarm rate is $\leq \alpha$. If so, we keep it for further consideration; if not, we discard it. Among those with acceptable false alarm rates, then, we ask "which classifier has the lowest false negative rate, the highest β?" This is the one we select.

Notice that this solves both problems with weighting. We don't have to pick a weight for the two errors; we just have to say what *rate* of false positives α we're willing to accept. There are many situations where this will be easier to do than to fix on a relative cost. Second, the rates α and β are properties of the *conditional* distributions of the variables, $\Pr(X|Y)$. If those conditional distributions stay they same but the proportions of the classes change, then the error rates are unaffected. Thus, training the classifier with a different mix of cases than we'll encounter in the future is not an issue.

Unfortunately, I don't know of any R implementation of Neyman-Pearson learning; it wouldn't be hard, I think, but goes beyond one problem set at this level.

13.4 Further Reading

The classic book on prediction trees, which basically introduced them into statistics and data mining, is Breiman *et al.* (1984). Chapter three in Berk (2008) is clear, easy to follow, and draws heavily on Breiman et al. Another very good chapter is the one on trees in Ripley (1996), which is especially useful for us because Ripley wrote the `tree` package. (The whole book is strongly recommended.) There is another tradition of trying to learn tree-structured models which comes out of artificial intelligence and inductive logic; see Mitchell (1997).

The clearest explanation of the Neyman-Pearson approach to hypothesis testing I have ever read is that in Reid (1982), which is one of the books which made me decide to learn statistics.

Exercises

13.1 Repeat the analysis of the California house-price data with the Pennsylvania data from Problem Set 10.

13.2 Explain why, for a *fixed* partition, a regression tree is a linear smoother.

13.3 Suppose that we see each of k classes n_i times, with $\sum_{i=1}^{k} n_i = n$. The maximum likelihood estimate of the probability of the ith class is $\widehat{p}_i = n_i/n$. Suppose that instead we use the estimates

$$\tilde{p}_i = \frac{n_i + 1}{\sum_{j=1}^{k} (n_j + 1)} \tag{13.9}$$

This estimator goes back to Laplace, who called it the "rule of succession".

1. Show that $\sum_i^k \tilde{p}_i = 1$, no matter what the sample is.
2. Show that if $\widehat{p} \to p$ as $n \to \infty$, then $\tilde{p} \to p$ as well.

3. Using the result of the previous part, show that if we observe an IID sample, that $\tilde{p} \to p$, i.e., that $\tilde{p}$ is a consistent estimator of the true distribution.
4. Does $\tilde{p} \to p$ imply $\widehat{p} \to p$?
5. Which of these properties still hold if the +1s in the numerator and denominator are replaced by $+d$ for an arbitrary $d > 0$?

13.4 *Fun with Laplace's rule of succession: will the Sun rise tomorrow?* One illustration Laplace gave of the probability estimator in Eq. 13.9 was the following. Suppose we know, from written records, that the Sun has risen in the east every day for the last 4000 years.[7]

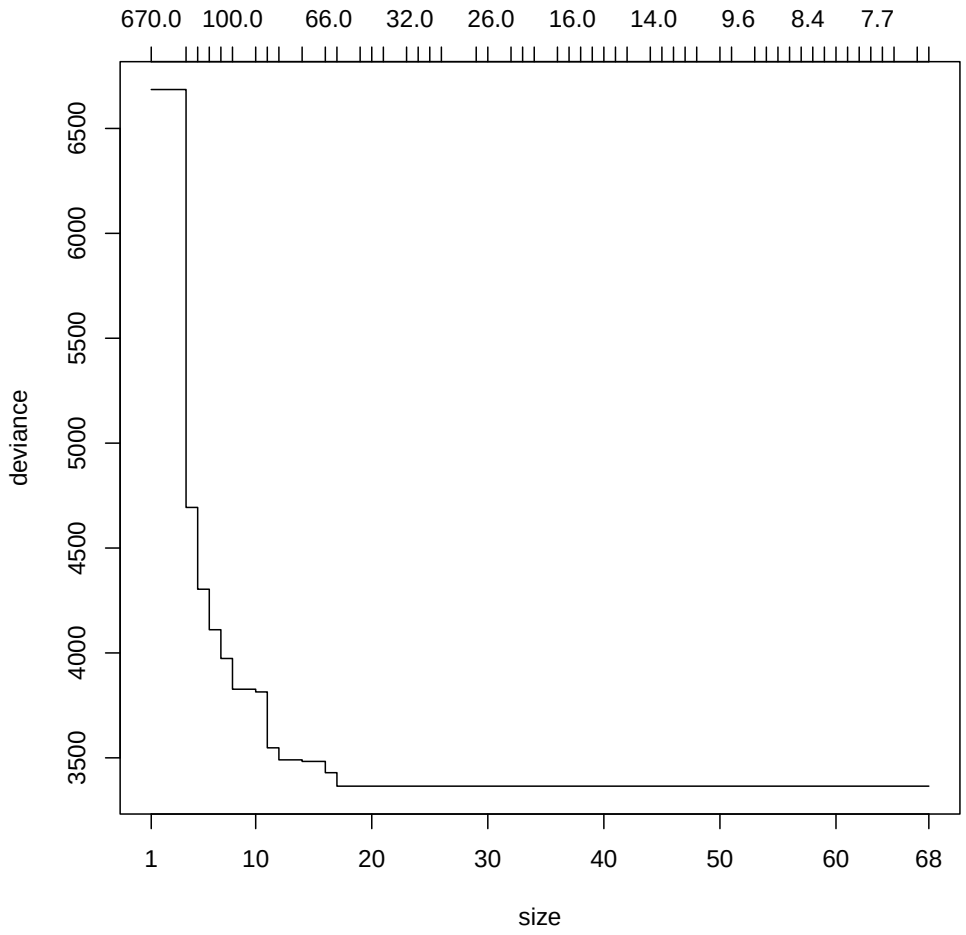
1. Calculate the probability of the event "the Sun will rise in the east tomorrow", using Eq. 13.9. You may take the year as containing 365.256 days.
2. Calculate the probability that the Sun will rise in the east *every day for the next four thousand years*, assuming this is an IID event each day. Is this a reasonable assumption?
3. Calculate the probability of the event "the Sun will rise in the east every day for four thousand years" directly from Eq. 13.9, treating *that* as a single event. Why does your answer here not agree with that of part (b)?

(Laplace did not, of course, base his belief that the Sun will rise in the morning on such calculations; besides everything else, he was the world's expert in celestial mechanics! But this shows a problem with the "rule of succession".)

13.5 It's reasonable to wonder why we should measure the complexity of a tree by just the number of leaves it has, rather than by the total number of nodes. Show that for a binary tree, with $|T|$ leaves, the total number of nodes (including the leaves) is $2|T| - 1$. (Thus, controlling the number of leaves is equivalent to controlling the number of nodes.)

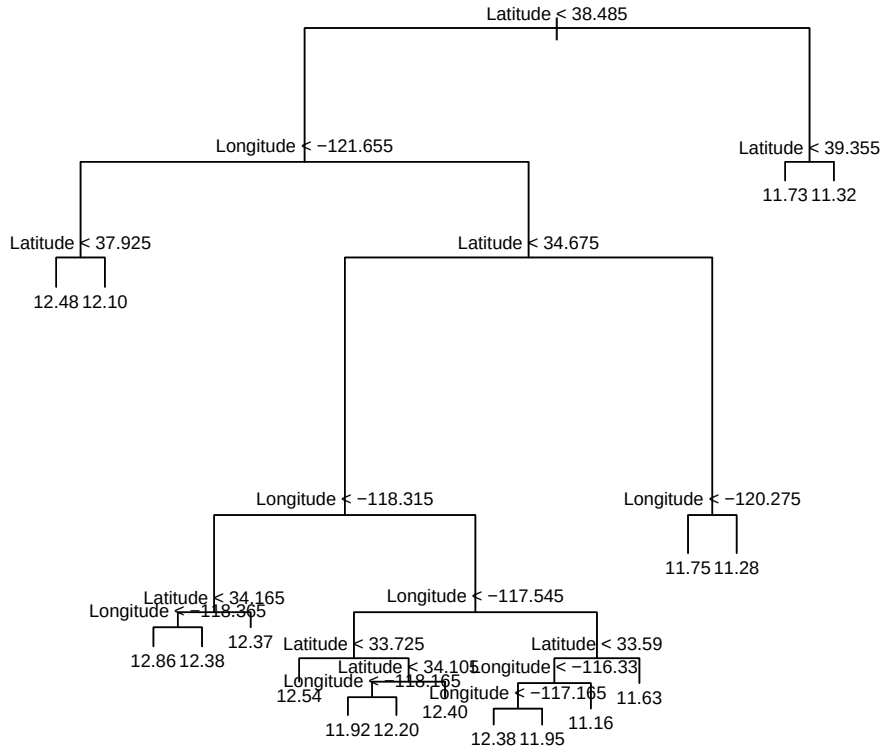
13.6 Show that, when all the off-diagonal elements of L_{ij} (from §13.3.3.2) are equal (and positive!), the best class to predict is always the most probable class .

[7] Laplace was thus ignoring people who live above the Artic circle, or below the Antarctic circle. The latter seems particularly unfair, because so many of them are scientists.



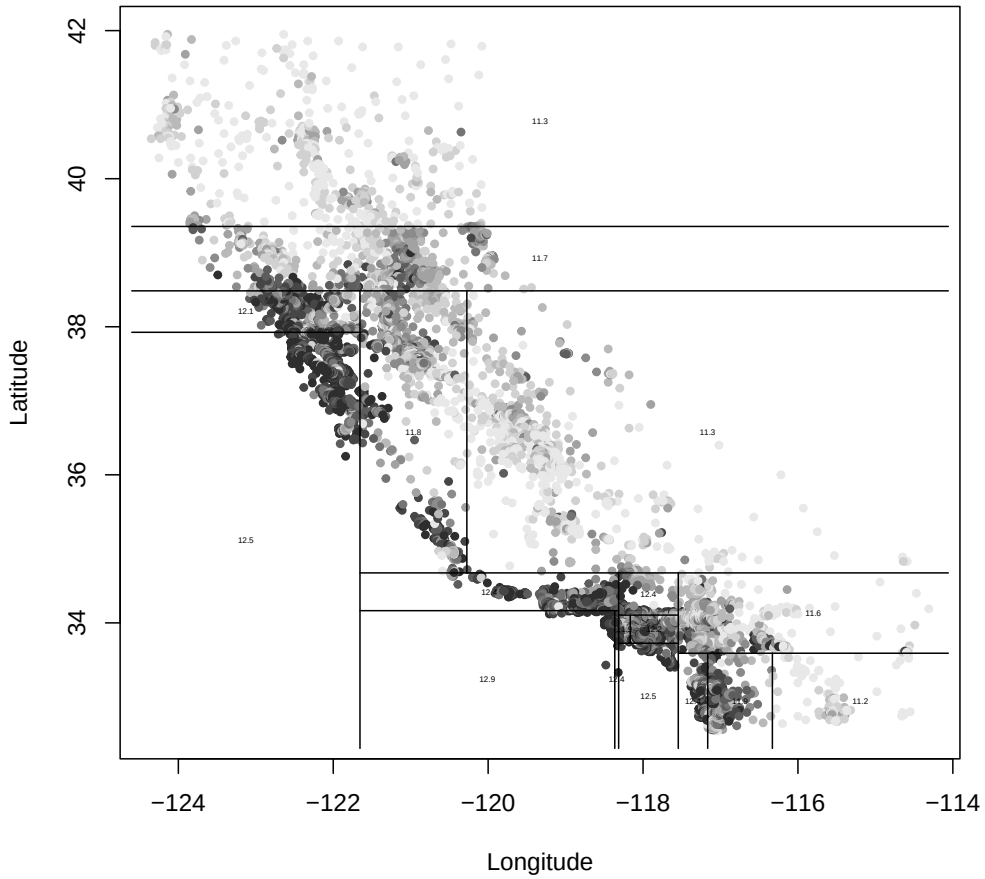

```
treefit2.cv <- cv.tree(treefit2)
plot(treefit2.cv)
```

Figure 13.8 Size (horizontal axis) versus cross-validated sum of squared errors (vertical axis) for successive prunings of the `treefit2` model. (The upper scale on the horizontal axis refers to the "cost/complexity" penalty. The idea is that the pruning minimizes (total error) + λ(complexity) for a certain value of λ, which is what's shown on that scale. Here complexity is taken to just be the number of leaves in the tree, i.e., its size (though sometimes other measures of complexity are used). λ then acts as a Lagrange multiplier (§D.3.2) which enforces a constraint on the complexity of the tree. See Ripley (1996, §7.2, pp. 221–226) for details.




```
opt.trees <- which(treefit2.cv$dev == min(treefit2.cv$dev))
best.leaves <- min(treefit2.cv$size[opt.trees])
treefit2.pruned <- prune.tree(treefit2, best = best.leaves)
plot(treefit2.pruned)
text(treefit2.pruned, cex = 0.75)
```

Figure 13.9 `treefit2`, after being pruned by ten-fold cross-validation.




```
plot(calif$Longitude, calif$Latitude, col = grey(10:2/11)[cut.prices], pch = 20,
    xlab = "Longitude", ylab = "Latitude")
partition.tree(treefit2.pruned, ordvars = c("Longitude", "Latitude"), add = TRUE,
    cex = 0.3)
```

Figure 13.10 `treefit2.pruned`'s partition of California. Compare to Figure 13.5.

Part II

Distributions and Latent Structure

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

14

Estimating Distributions and Densities

We have spent a lot of time looking at how to estimate expectations (which is regression). We have also seen how to estimate variances, by turning it into a problem about expectations. We could extend the same methods to looking at higher moments — if you need to find the conditional skewness or kurtosis functions[1], you can tackle that in the same way as finding the conditional variance. But what if we want to look at the whole distribution?

You've already seen one solution to this problem in earlier statistics courses: posit a parametric model for the density (Gaussian, Student's t, exponential, gamma, beta, Pareto, ...) and estimate the parameters. Maximum likelihood estimates are generally consistent and efficient for such problems. None of this changes when the distributions are multivariate. But suppose you don't have any particular parametric density family in mind, or want to check one — how could we estimate a probability distribution non-parametrically?

14.1 Histograms Revisited

For most of you, making a histogram was probably one of the first things you learned how to do in intro stats (if not before). This is a simple way of estimating a distribution: we split the sample space up into bins, count how many samples fall into each bin, and then divide the counts by the total number of samples. If we hold the bins fixed and take more and more data, then by the law of large numbers we anticipate that the relative frequency for each bin will converge on the bin's probability.

So far so good. But one of the things you learned in intro stats was also to work with probability *density* functions, not just probability *mass* functions. Where do we get pdfs? Well, one thing we could do is to take our histogram estimate, and then say that the probability density is uniform within each bin. This gives us a piecewise-constant estimate of the density.

Unfortunately, this isn't going to work — isn't going to converge on the true pdf — unless we can shrink the bins of the histogram as we get more and more data. To see this, think about estimating the pdf when the data comes from any of the standard distributions, like an exponential or a Gaussian. We can approximate the true pdf $f(x)$ to arbitrary accuracy by a piecewise-constant density (indeed,

[1] When you find out what the kurtosis is good for, be sure to tell the world.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

that's what happens every time we plot it on our screens), but, for a fixed set of bins, we can only come so close to the true, continuous density.

This reminds us of our old friend the bias-variance trade-off, and rightly so. If we use a large number of very small bins, the minimum bias in our estimate of any density becomes small, but the variance in our estimates grows. (Why does variance increase?) To make some use of this insight, though, there are some things we need to establish first.

- Is learning the whole distribution non-parametrically even feasible?
- How can we measure error so deal with the bias-variance trade-off?

14.2 "The Fundamental Theorem of Statistics"

Let's deal with the first point first. In principle, something even dumber than shrinking histograms will work to learn the whole distribution. Suppose we have one-dimensional samples $x_1, x_2, \ldots x_n$ with a common cumulative distribution function F. The **empirical cumulative distribution function** on n samples, $\tilde{F}_n(a)$ is

$$\tilde{F}_n(a) \equiv \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}_{(-\infty,a])}(x_i) \tag{14.1}$$

In words, this is just the fraction of the samples which are $\leq a$. Then the **Glivenko-Cantelli theorem** says

$$\max_a |\tilde{F}_n(a) - F(a)| \to 0 \tag{14.2}$$

So the empirical CDF converges to the true CDF everywhere; the maximum gap between the two of them goes to zero. Pitman (1979) calls this the "fundamental theorem of statistics", because it says we can learn distributions just by collecting enough data.[2] The same kind of result also holds for the CDFs of higher-dimensional vectors.

If the Glivenko-Cantelli theorem is so great, why aren't we just content with

[2] There are some interesting aspects to the theorem which are tangential to what we'll need, so I will stick them in this footnote. These hinge on the max in the statement of the theorem. For any one, fixed value of a, that $|\tilde{F}_n(a) - F(a)| \to 0$ is just an application of the law of large numbers. The extra work Glivenko and Cantelli did was to show that this held for *infinitely many* values of a at once, so that even if we focus on the biggest gap between the estimate and the truth, that still shrinks with n. Here's a sketch, with no details. Fix an $\epsilon > 0$; first show that there is some *finite* set of points on the line, call them $b_1, \ldots b_{m(\epsilon)}$, such that, for any a, $|\tilde{F}_n(a) - \tilde{F}_n(b_i)| < \epsilon$ and $|F(a) - F(b_i)| < \epsilon$ for *some* b_i. Next, show that, for large enough n, $|F(b_i) - \tilde{F}_n(b_i)| < \epsilon$ for all the b_i simultaneously. (This follows from the law of large numbers and the fact that $m(\epsilon)$ is finite.) Finally, use the triangle inequality to conclude that, for large enough n, $\max_a |\tilde{F}_n(a) - F(a)| < 3\epsilon$. Since ϵ can be made arbitrarily small, the Glivenko-Cantelli theorem follows. This general strategy — combining pointwise convergence theorems with approximation arguments — forms the core of what's called **empirical process theory**, which underlies the consistency of basically all the non-parametric procedures we've seen. If this line of thought is at all intriguing, the closest thing to a gentle introduction is Pollard (1989). (If you know enough to object that I should have been writing sup instead of max, you know enough to make the substitution for yourself.)

the empirical CDF? Sometimes we are, but it inconveniently doesn't give us a probability *density*. Suppose that $x_1, x_2, \ldots x_n$ are sorted into increasing order. What probability does the empirical CDF put on the interval (x_i, x_{i+1})? Clearly, zero. (Whereas the interval $[x_i, x_{i+1}]$ gets probability $2/n$.) This *could* be right, but we have centuries of experience now with probability distributions, and this tells us that *pretty often* we can expect to find some new samples between our old ones. So we'd like to get a *non-zero* density between our observations.

Using a uniform distribution within each bin of a histogram doesn't have this issue, but it does leave us with the problem of picking where the bins go and how many of them we should use. Of course, there's nothing magic about keeping the bin size the same and letting the number of points in the bins vary; we could equally well pick bins so they had equal counts.[3] So what should we do?

14.3 Error for Density Estimates

Our first step is to get clear on what we mean by a "good" density estimate. There are three leading ideas:

1. $\int \left(f(x) - \hat{f}(x)\right)^2 dx$ should be small: the squared deviation from the true density should be small, averaging evenly over all space.
2. $\int |f(x) - \hat{f}(x)| dx$ should be small: minimize the average *absolute*, rather than squared, deviation.
3. $\int f(x) \log \frac{f(x)}{\hat{f}(x)} dx$ should be small: the average log-likelihood ratio should be kept low.

Option (1) is reminiscent of the MSE criterion we've used in regression. Option (2) looks at what's called the L_1 or **total variation** distance between the true and the estimated density. It has the nice property that $\frac{1}{2}\int |f(x) - \hat{f}(x)| dx$ is exactly the maximum error in our estimate of the probability of *any* set. Unfortunately it's a bit tricky to work with, so we'll skip it here. (But see Devroye and Lugosi (2001)). Finally, minimizing the log-likelihood ratio is intimately connected to maximizing the likelihood. We will come back to this (§14.6), but, like most texts on density estimation, we will give more attention to minimizing (1), because it's mathematically tractable.

Notice that

$$\int \left(f(x) - \hat{f}(x)\right)^2 dx = \int f^2(x) dx - 2\int \hat{f}(x) f(x) dx + \int \hat{f}^2(x) dx \tag{14.3}$$

[3] A specific idea for how to do this is sometimes called a $k-d$ tree. We have d random variables and want a joint density for all of them. Fix an ordering of the variables Start with the first variable, and find the thresholds which divide it into k parts with equal counts. (Usually but not always $k = 2$.) Then sub-divide each part into k equal-count parts on the *second* variable, then sub-divide each of those on the third variable, etc. After splitting on the dth variable, go back to splitting on the first, until no further splits are possible. With n data points, it takes about $\log_k n$ splits before coming down to individual data points. Each of these will occupy a cell of some volume. Estimate the density on that cell as one over that volume. Of course it's not strictly necessary to keep refining all the way down to single points.

The first term on the right hand side doesn't depend on the estimate $\hat{f}(x)$ at all, so we can ignore it for purposes of optimization. The third one only involves $\hat{f}$, and is just an integral, which we can do numerically. That leaves the middle term, which involves both the true and the estimated density; we can approximate it by

$$-\frac{2}{n}\sum_{i=1}^{n}\hat{f}(x_i) \tag{14.4}$$

The reason we can do this is that, by the Glivenko-Cantelli theorem, integrals over the true density are approximately equal to sums over the empirical distribution.

So our final error measure is

$$-\frac{2}{n}\sum_{i=1}^{n}\hat{f}(x_i) + \int \hat{f}^2(x)dx \tag{14.5}$$

In fact, this error measure does *not* depend on having one-dimension data; we can use it in any number of dimensions.[4] For purposes of cross-validation (you knew that was coming, right?), we can estimate $\hat{f}$ on the training set, and then restrict the sum to points in the testing set.

14.3.1 Error Analysis for Histogram Density Estimates

We now have the tools to do most of the analysis of histogram density estimation. (We'll do it in one dimension for simplicity.) Choose our favorite location x, which lies in a bin whose boundaries are x_0 and $x_0 + h$. We want to estimate the density at x, and this is

$$\hat{f}_n(x) = \frac{1}{h}\frac{1}{n}\sum_{i=1}^{n}\mathbf{1}_{(x_0,x_0+h]}(x_i) \tag{14.6}$$

Let's call the sum, the number of points in the bin, b. It's a random quantity, $B \sim \text{Binomial}(n,p)$, where p is the true probability of falling into the bin, $p = F(x_0+h) - F(x_0)$. The mean of B is np, and the variance is $np(1-p)$, so

$$\mathbb{E}\left[\hat{f}_n(x)\right] = \frac{1}{nh}\mathbb{E}[B] \tag{14.7}$$

$$= \frac{n[F(x_0+h) - F(x_0)]}{nh} \tag{14.8}$$

$$= \frac{F(x_0+h) - F(x_0)}{h} \tag{14.9}$$

[4] Admittedly, in high-dimensional spaces, doing the final integral can become numerically challenging.

and the variance is

$$\mathbb{V}\left[\hat{f}_n(x)\right] = \frac{1}{n^2h^2}\mathbb{V}[B] \tag{14.10}$$

$$= \frac{n[F(x_0+h)-F(x_0)][1-F(x_0+h)+F(x_0)]}{n^2h^2} \tag{14.11}$$

$$= \mathbb{E}\left[\hat{f}_n(x)\right]\frac{1-F(x_0+h)+F(x_0)}{nh} \tag{14.12}$$

If we let $h \to 0$ as $n \to \infty$, then

$$\mathbb{E}\left[\hat{f}_h(x)\right] \to \lim_{h\to 0}\frac{F(x_0+h)-F(x_0)}{h} = f(x_0) \tag{14.13}$$

since the pdf is the derivative of the CDF. But since x is between x_0 and x_0+h, $f(x_0) \to f(x)$. So if we use smaller and smaller bins as we get more data, the histogram density estimate is unbiased. We'd also like its variance to shrink as the same grows. Since $1-F(x_0+h)+F(x_0) \to 1$ as $h \to 0$, to get the variance to go away we need $nh \to \infty$.

To put this together, then, our first conclusion is that histogram density estimates will be consistent when $h \to 0$ but $nh \to \infty$ as $n \to \infty$. The bin-width h needs to shrink, but slower than n^{-1}.

At what rate should it shrink? Small h gives us low bias but (as you can verify from the algebra above) high variance, so we want to find the trade-off between the two. One can calculate the bias at x from our formula for $\mathbb{E}\left[\hat{f}_h(x)\right]$ through a somewhat lengthy calculus exercise, analogous to what we did for kernel smoothing in Chapter 4[5]; the upshot is that the integrated squared bias is

$$\int\left(f(x)-\mathbb{E}\left[\hat{f}_h(x)\right]\right)^2 dx = \frac{h^2}{12}\int\left(f'(x)\right)^2 dx + o(h^2) \tag{14.14}$$

We already got the variance at x, and when we integrate that over x we find

$$\int \mathbb{V}\left[\hat{f}_h(x)\right] dx = \frac{1}{nh} + o(n^{-1}) \tag{14.15}$$

So the total integrated squared error is

$$\text{ISE} = \frac{h^2}{12}\int\left(f'(x)\right)^2 dx + \frac{1}{nh} + o(h^2) + o(n^{-1}) \tag{14.16}$$

Differentiating this with respect to h and setting it equal to zero, we get

$$\frac{h_{\text{opt}}}{6}\int\left(f'(x)\right)^2 dx = \frac{1}{nh_{\text{opt}}^2} \tag{14.17}$$

$$h_{\text{opt}} = \left(\frac{6}{\int\left(f'(x)\right)^2 dx}\right)^{1/3} n^{-1/3} = O(n^{-1/3}) \tag{14.18}$$

[5] You need to use the intermediate value theorem multiple times; see for instance Wasserman (2006, sec. 6.8).

So we need narrow bins if the density changes rapidly ($\int (f'(x))^2 dx$ is large), and wide bins if the density is relatively flat. No matter how rough the density, the bin width should shrink like $O(n^{-1/3})$. Plugging that rate back into the equation for the ISE, we see that it is $O(n^{-2/3})$.

It turns out that if we pick h by cross-validation, then we attain this optimal rate in the large-sample limit. By contrast, if we *knew* the correct parametric form and just had to estimate the parameters, we'd typically get an error decay of $O(n^{-1})$. This is substantially faster than histograms, so it would be nice if we could make up some of the gap, without having to rely on parametric assumptions.

14.4 Kernel Density Estimates

It turns out that one can improve the convergence rate, as well as getting smoother estimates, by using kernels. The **kernel density estimate** is

$$\widehat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} \frac{1}{h} K\left(\frac{x - x_i}{h}\right) \tag{14.19}$$

where K is a kernel function such as we encountered when looking at kernel regression. (The factor of $1/h$ inside the sum is so that $\widehat{f}_h$ will integrate to 1; we could have included it in both the numerator and denominator of the kernel regression formulae, but then it would've just canceled out.) As before, h is the **bandwdith** of the kernel. We've seen typical kernels in things like the Gaussian. One advantage of using them is that they give us a smooth density everywhere, unlike histograms, and in fact we can even use them to estimate the derivatives of the density, should that be necessary.[6]

14.4.1 Analysis of Kernel Density Estimates

How do we know that kernels will in fact work? Well, let's look at the mean and variance of the kernel density estimate at a particular point x, and use Taylor's

[6] The advantage of histograms is that they're computationally and mathematically simpler.

theorem on the density.

$$\mathbb{E}\left[\widehat{f}_h(x)\right] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[\frac{1}{h}K\left(\frac{x-X_i}{h}\right)\right] \tag{14.20}$$

$$= \mathbb{E}\left[\frac{1}{h}K\left(\frac{x-X}{h}\right)\right] \tag{14.21}$$

$$= \int \frac{1}{h}K\left(\frac{x-t}{h}\right) f(t)dt \tag{14.22}$$

$$= \int K(u)f(x-hu)du \tag{14.23}$$

$$= \int K(u)\left[f(x) - huf'(x) + \frac{h^2u^2}{2}f''(x) + o(h^2)\right]du \tag{14.24}$$

$$= f(x) + \frac{h^2 f''(x)}{2}\int K(u)u^2du + o(h^2) \tag{14.25}$$

$$\tag{14.26}$$

because, by definition, $\int K(u)du = 1$ and $\int uK(u)du = 0$. If we call $\int K(u)u^2du = \sigma_K^2$, then the bias of the kernel density estimate is

$$\mathbb{E}\left[\widehat{f}_h(x)\right] - f(x) = \frac{h^2\sigma_K^2 f''(x)}{2} + o(h^2) \tag{14.27}$$

So the bias will go to zero if the bandwidth h shrinks to zero. What about the variance? Use Taylor's theorem again:

$$\mathbb{V}\left[\widehat{f}_h(x)\right] = \frac{1}{n}\mathbb{V}\left[\frac{1}{h}K\left(\frac{x-X}{h}\right)\right] \tag{14.28}$$

$$= \frac{1}{n}\left[\mathbb{E}\left[\frac{1}{h^2}K^2\left(\frac{x-X}{h}\right)\right] - \left(\mathbb{E}\left[\frac{1}{h}K\left(\frac{x-X}{h}\right)\right]\right)^2\right] \tag{14.29}$$

$$= \frac{1}{n}\left[\int \frac{1}{h^2}K^2\left(\frac{x-t}{h}\right)dt - \left[f(x) + O(h^2)\right]^2\right] \tag{14.30}$$

$$= \frac{1}{n}\left[\int \frac{1}{h}K^2(u)f(x-hu)du - f^2(x) + O(h^2)\right] \tag{14.31}$$

$$= \frac{1}{n}\left[\int \frac{1}{h}K^2(u)\left(f(x) - huf'(x)\right)du - f^2(x) + O(h)\right] \tag{14.32}$$

$$= \frac{f(x)}{hn}\int K^2(u)du + O(1/n) \tag{14.33}$$

This will go to zero if $nh \to \infty$ as $n \to \infty$. So the conclusion is the same as for histograms: h has to go to zero, but slower than $1/n$.

Since the expected squared error at x is the bias squared plus the variance,

$$\frac{h^4\sigma_K^4(f''(x))^2}{4} + \frac{f(x)}{hn}\int K^2(u)du + \text{small} \tag{14.34}$$

the expected integrated squared error is

$$\text{ISE} \approx \frac{h^4 \sigma_K^4}{4} \int (f''(x))^2 dx + \frac{\int K^2(u) du}{nh} \tag{14.35}$$

Differentiating with respect to h for the optimal bandwidth h_{opt}, we find

$$h_{\text{opt}}^3 \sigma_K^4 \int (f''(x))^2 dx = \frac{\int K^2(u) du}{n h_{\text{opt}}^2} \tag{14.36}$$

$$h_{\text{opt}} = \left(\frac{\int K^2(u) du}{\sigma_K^4 \int (f''(x))^2 dx} \right)^{1/5} n^{-1/5} = O(n^{-1/5}) \tag{14.37}$$

That is, the best bandwidth goes to zero like one over the fifth root of the number of sample points. Plugging this into Eq. 14.35, the best ISE $= O(n^{-4/5})$. This is better than the $O(n^{-2/3})$ rate of histograms, but still includes a penalty for having to figure out what kind of distribution we're dealing with. Remarkably enough, using cross-validation to pick the bandwidth gives near-optimal results.[7]

As an alternative to cross-validation, or at least a starting point, one can use Eq. 14.37 to show that the optimal bandwidth for using a Gaussian kernel to estimate a Gaussian distribution is $1.06 \sigma n^{-1/5}$, with σ being the standard deviation of the Gaussian. This is sometimes called the **Gaussian reference rule** or the **rule-of-thumb** bandwidth. When you call `density` in R, this is basically what it does.

Yet another technique is the **plug-in method**. Eq. 14.37 calculates the optimal bandwidth from the second derivative of the true density. This doesn't help if we don't know the density, but it becomes useful if we have an initial density estimate which isn't too bad. In the plug-in method, we start with an initial bandwidth (say from the Gaussian reference rule) and use it to get a preliminary estimate of the density. Taking that crude estimate and "plugging it in" to Eq. 14.37 gives us a new bandwidth, and we re-do the kernel estimate with that new bandwidth. Iterating this a few times is optional but not uncommon.

14.4.2 Joint Density Estimates

The discussion and analysis so far has been focused on estimating the distribution of a one-dimensional variable. Just as kernel regression can be done with multiple input variables (§4.3), we can make kernel density estimates of joint distributions. We simply need a kernel for the vector:

$$\widehat{f}(\vec{x}) = \frac{1}{n} \sum_{i=1}^{n} K(\vec{x} - \vec{x_i}) \tag{14.38}$$

[7] Substituting Eq. 14.37 into Eq. 14.35 gives a squared error of $1.25 n^{-4/5} \sigma_K^{4/5} \left(\int (f''(x))^2 dx \right)^{1/5} \left(\int K^2(u) du \right)^{4/5}$. The only two parts of this which depend on the kernel are σ_K and $\int K^2(u) du$. This is the source of the (correct) folklore that the choice of kernel is less important than the choice of bandwidth.

One could use any multivariate distribution as the kernel (provided it is centered and has finite covariance). Typically, however, just as in smoothing, one uses a product kernel, i.e., a product of one-dimensional kernels,

$$K(\vec{x} - \vec{x_i}) = K_1(x^1 - x_i^1)K_2(x^2 - x_i^2)\ldots K_p(x^p - x_i^p) \ , \tag{14.39}$$

Doing this requires a bandwidth for each coordinate, so the over-all form of the joint PDF estimate is

$$\widehat{f}(\vec{x}) = \frac{1}{n\prod_{j=1}^{p} h_j} \sum_{i=1}^{n} \prod_{j=1}^{d} K_j\left(\frac{x^j - x_i^j}{h_j}\right) \tag{14.40}$$

Going through a similar analysis for p-dimensional data shows that the ISE goes to zero like $O(n^{-4/(4+p)})$, and again, if we use cross-validation to pick the bandwidths, asymptotically we attain this rate. Unfortunately, if p is large, this rate becomes very slow — for instance, if $p = 24$, the rate is $O(n^{-1/7})$. There is simply no universally good way to learn *arbitrary* high-dimensional distributions. This is the same "curse of dimensionality" we saw in regression (§8.3). The fundamental problem is that in high dimensions, there are just too different possible distributions which are too hard to tell apart.

Evading the curse of dimensionality for density estimation needs some special assumptions. Parametric models make the very strong assumption that we know exactly what the distribution function looks like, and we just need to fill in a few constants. It's potentially less drastic to hope the distribution has some sort of special structure we can exploit, and most of the rest of Part II will be about searching for various sorts of useful structure[8]. If none of these options sound appealing, or plausible, we've got little alternative but to accept a very slow convergence of density estimates.

14.4.3 Categorical and Ordered Variables

Estimating probability mass functions with discrete variables can be straightforward: there are only a finite number of values, and so one just counts how often they occur and takes the relative frequency. If one has a discrete variable X and a continuous variable Y and one wants a joint distribution, one could just get a separate density for Y for each value of x, and tabulate the probabilities for x.

In principle, this will work, but it can be practically awkward if the number of levels for the discrete variable is large compared to the number of samples. Moreover, for the joint distribution problem, it has us estimating completely separate distributions for Y for every x, without any sharing of information between them. It would seem more plausible to smooth those distributions towards each others. To do this, we need kernels for discrete variables.

Several sets of such kernels have been proposed. The most straightforward,

[8] As Wiener (1956), the reason the ability to do nonparametric estimation doesn't make scientific theories redundant is that good theories usefully constrain the distributions we're searching for, and tell us what structures to look for.

however, are the following. If X is a categorical, unordered variable with c possible values, then, for $0 \leq h < 1$,

$$K(x_1, x_2) = \begin{cases} 1-h & x_1 = x_2 \\ h/(c-1) & x_1 \neq x_2 \end{cases} \tag{14.41}$$

is a valid kernel. For an ordered x,

$$K(x_1, x_2) = \binom{c}{|x_1 - x_2|} h^{|x_1 - x_2|} (1-h)^{c - |x_1 - x_2|} \tag{14.42}$$

where $|x_1 - x_2|$ should be understood as just how many levels apart x_1 and x_2 are. As $h \to 0$, both of these become indicators, and return us to simple relative frequency counting. Both of these are implemented in `np`.

14.4.4 Practicalities

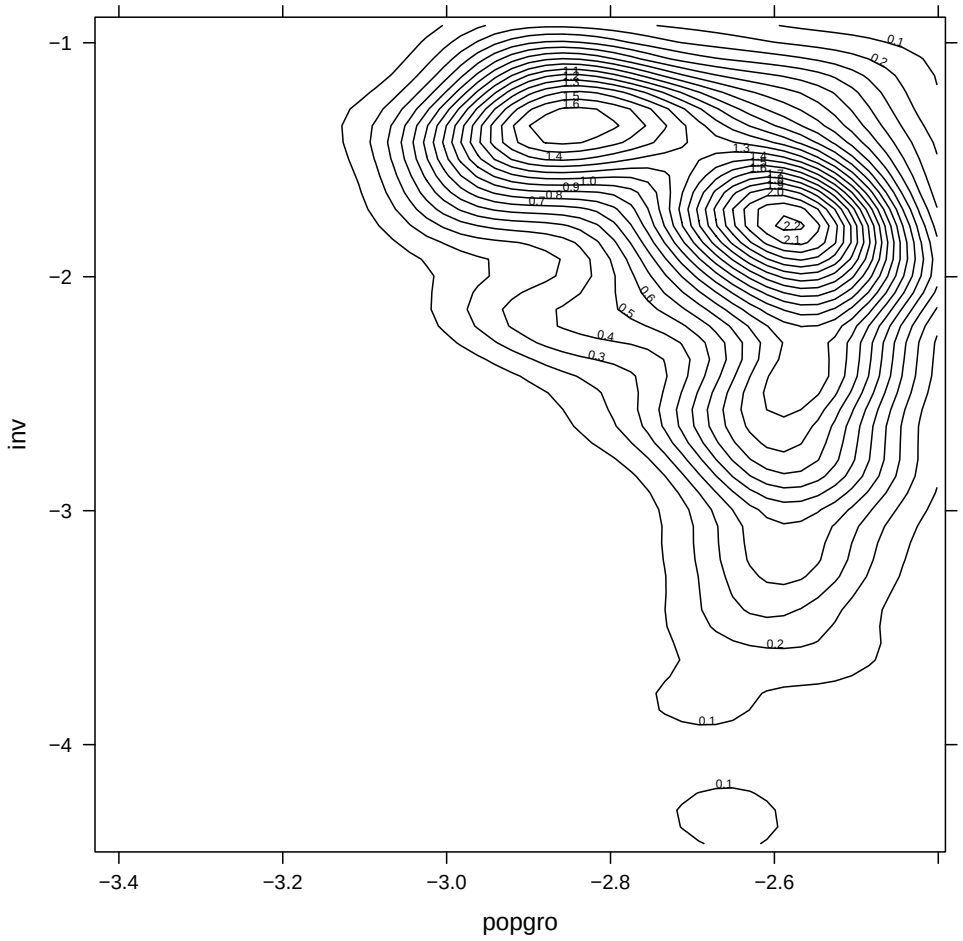
The standard R function `density` implements one-dimensional kernel density estimation, defaulting to Gaussian kernels with the rule-of-thumb bandwidth. There are some options for doing cleverer bandwidth selection, including a plug-in rule. (See the help file.)

For more sophisticated methods, and especially for more dimensions, you'll need to use other packages. The `np` package estimates joint densities using the `npudens` function. (The `u` is for "unconditional".) This has the same sort of automatic bandwidth selection as `npreg`, using cross-validation. Other packages which do kernel density estimation include `KernSmooth` and `sm`.

14.4.5 Kernel Density Estimation in R: An Economic Example

The data set `oecdpanel`, in the `np` library, contains information about much the same sort of variables at the Penn World Tables data you worked with in the homework, over much the same countries and years, but with some of the variables pre-transformed, with identifying country information removed, and slightly different data sources. See `help(oecdpanel)` for details.

Here's an example of using `npudens` with variables from the `oecdpanel` data set, from problem set 11. We'll look at the joint density of `popgro` (the logarithm of the population growth rate) and `inv` (the logarithm of the investment rate). Figure 14.1 illustrates how to call the command, and a useful trick where we get `np`'s plotting function to do our calculations for us, but then pass the results to a different graphics routine. (See `help(npplot)`.) The distribution we get has two big modes, one at a comparatively low population growth rate (≈ -2.9 — remember this is logged so it's not actually a shrinking population) and high investment (≈ -1.5), and the other at a lower rate of investment (≈ -2) and higher population growth (≈ -2.6). There is a third, much smaller mode at high population growth (≈ -2.7) and very low investment (≈ -4).




```
data(oecdpanel)
popinv <- npudens(~popgro + inv, data = oecdpanel)
fhat <- plot(popinv, plot.behavior = "data")$d1
library(lattice)
contourplot(fhat$dens ~ fhat$eval$Var1 * fhat$eval$Var2, cuts = 20, xlab = "popgro",
    ylab = "inv", labels = list(cex = 0.5))
```

Figure 14.1 Gaussian kernel estimate of the joint distribution of logged population growth rate (`popgro`) and investment rate (`inv`). Notice that `npudens` takes a formula, but that there is no dependent variable on the left-hand side of the ∼. With objects produced by the `np` library, one can give the plotting function the argument `plot.behavior` — the default is `plot`, but if it's set to `data` (as here), it calculates all the information needed to plot and returns a separate set of objects, which can be plotted in other functions. (The value `plot-data` does both.) See `help(npplot)` for more.

14.5 Conditional Density Estimation

In addition to estimating marginal and joint densities, we will often want to get conditional densities. The most straightforward way to get the density of Y given X, $f_{Y|X}(y \mid x)$, is

$$\widehat{f}_{Y|X}(y \mid x) = \frac{\widehat{f}_{X,Y}(x,y)}{\widehat{f}_X(x)} \tag{14.43}$$

i.e., to estimate the joint and marginal densities and divide one by the other.

To be concrete, let's suppose that we are using a product kernel to estimate the joint density, and that the marginal density is consistent with it:

$$\widehat{f}_{X,Y}(x,y) = \frac{1}{nh_X h_Y} \sum_{i=1}^{n} K_X\left(\frac{x - x_i}{h_X}\right) K_Y\left(\frac{y - y_i}{h_Y}\right) \tag{14.44}$$

$$\widehat{f}_X(x) = \frac{1}{nh_X} \sum_{i=1}^{n} K_X\left(\frac{x - x_i}{h_X}\right) \tag{14.45}$$

Thus we need to pick two bandwidths, h_X and h_Y, one for each variable.

This might seem like a solved problem — we just use cross-validation to find h_X and h_Y so as to minimize the integrated squared error for $\widehat{f}_{X,Y}$, and then plug in to Equation 14.43. However, this is a bit hasty, because the optimal bandwidths for the *joint* density are not necessarily the optimal bandwidths for the *conditional* density. An extreme but easy to understand example is when Y is actually independent of X. Since the density of Y given X is just the density of Y, we'd be best off just ignoring X by taking $h_X = \infty$. (In practice, we'd just use a very big bandwidth.) But if we want to find the joint density, we would not want to smooth X away completely like this.

The appropriate integrated squared error measure for the conditional density is

$$\int dx f_X(x) \int dy \Big(f_{Y|X}(y \mid x) - \widehat{f}_{Y|X}(y \mid x)\Big)^2 \tag{14.46}$$

and this is what we want to minimize by picking h_X and h_Y. The cross-validation goes as usual.

One nice, and quite remarkable, property of cross-validation for conditional density estimation is that it can detect and exploit conditional independence. Say that $X = (U, V)$, and that Y is independent of U given V — symbolically, $Y \perp\!\!\!\perp U \mid V$. Then $f_{Y|U,V}(y \mid u, v) = f_{Y|V}(y \mid v)$, and we should just ignore U in our estimation of the conditional density. It turns out that when cross-validation is used to pick bandwidths for conditional density estimation, $\widehat{h_U} \to \infty$ when $Y \perp\!\!\!\perp U \mid V$, but not otherwise (Hall *et al.*, 2004). In other words, cross-validation will automatically detect which variables are irrelevant, and smooth them away.

14.5.1 Practicalities and a Second Example

The `np` package implements kernel conditional density estimation through the function `npcdens`. The syntax is pretty much exactly like that of `npreg`, and indeed we can think of estimating the conditional density as a sort of regression, where the dependent variable is actually a distribution.

To give a concrete example, let's look at how the distribution of countries' population growth rates has changed over time, using the `oecdpanel` data (Figure 14.2). The selected bandwidth for `year` is 10, while that for `popgro` is 0.048. (Note that `year` is being treated as a continuous variable.)

You can see from the figure that the mode for population growth rates is towards the high end of observed values, but the mode is shrinking and becoming less pronounced over time. The distribution in fact begins as clearly bimodal, but the smaller mode at the lower growth rate turns into a continuous "shoulder". Over time, Figure 14.2 population growth rates tend to shrink, and the dispersion of growth rates narrows.

Let's expand on this point. One of the variables in `oecdpanel` is `oecd`, which is 1 for countries which are members of the Organization for Economic Cooperation and Development, and 0 otherwise. The OECD countries are basically the "developed" ones (stable capitalist democracies). We can include OECD membership as a conditioning variable for population growth (we need to use a categorical-variable kernel), and look at the combined effect of time and development (Figure 14.3).

What the figure shows is that OECD and non-OECD countries both have unimodal distributions of growth rates. The mode for the OECD countries has become sharper, but the value has decreased. The mode for non-OECD countries has also decreased, while the distribution has become more spread out, mostly by having more probability of lower growth rates. (These trends have continued since 1995.) In words, despite the widespread contrary impression, population growth has actually been slowing for decades in both rich and poor countries.

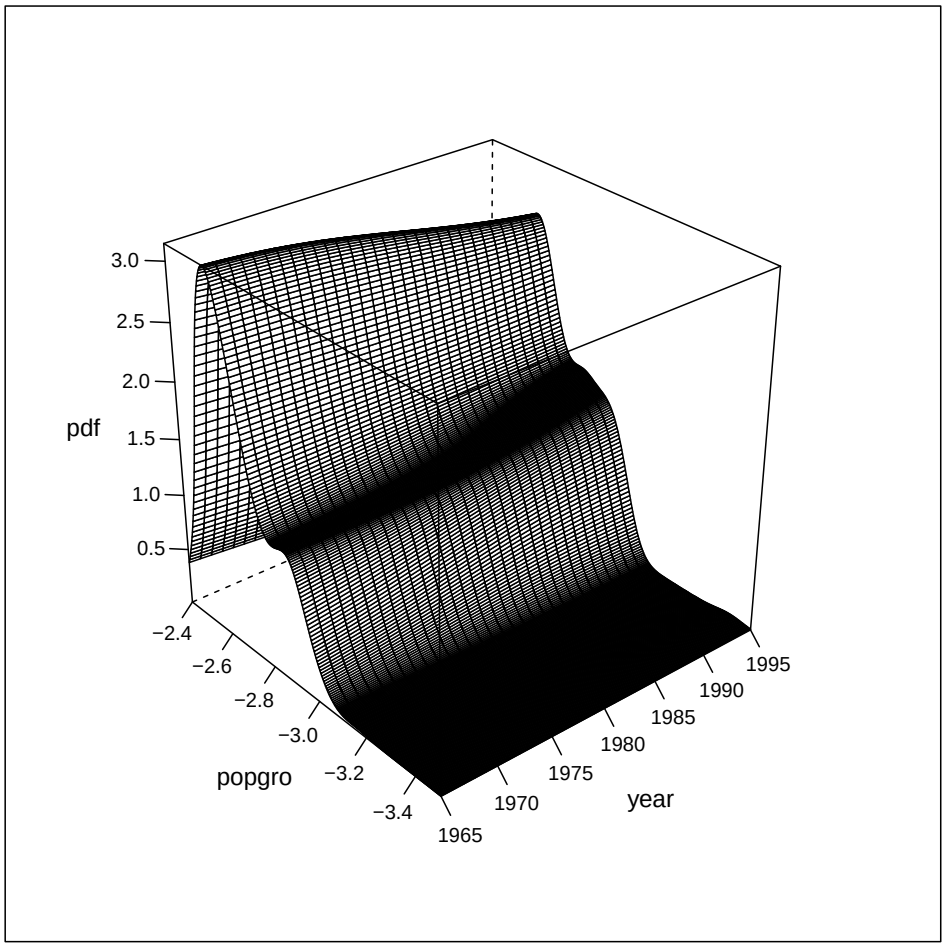
14.6 More on the Expected Log-Likelihood Ratio

I want to say just a bit more about the expected log-likelihood ratio $\int f(x) \log \frac{f(x)}{\widehat{f}(x)} dx$. More formally, this is called the **Kullback-Leibler divergence** or **relative entropy** of $\widehat{f}$ from f, and is also written $D(f\|\widehat{f})$. Let's expand the log ratio:

$$D(f\|\widehat{f}) = -\int f(x) \log \widehat{f}(x) dx + \int f(x) \log f(x) dx \tag{14.47}$$

The second term does not involve the density estimate, so it's irrelevant for purposes of optimizing over $\widehat{f}$. (In fact, we're just subtracting off the entropy of the true density.) Just as with the squared error, we could try approximating the integral with a sum:

$$\int f(x) \log \widehat{f}(x) dx \approx \frac{1}{n} \sum_{i=1}^{n} \log \widehat{f}(x_i) \tag{14.48}$$

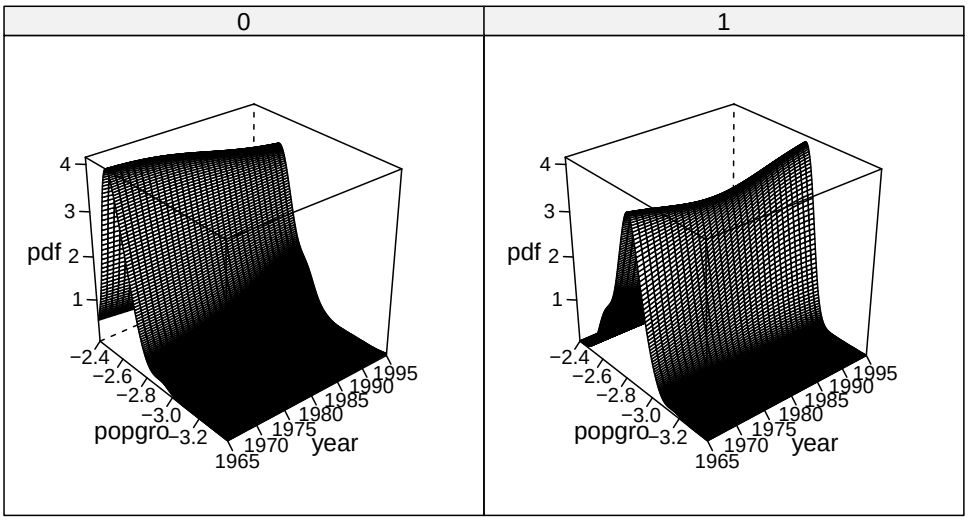



```
pop.cdens <- npcdens(popgro ~ year, data = oecdpanel)
plotting.grid <- expand.grid(year = seq(from = 1965, to = 1995, by = 1), popgro = seq(from = -3.5,
    to = -2.4, length.out = 300))
fhat <- predict(pop.cdens, newdata = plotting.grid)
wireframe(fhat ~ plotting.grid$year * plotting.grid$popgro, scales = list(arrows = FALSE),
    xlab = "year", ylab = "popgro", zlab = "pdf")
```

Figure 14.2 Conditional density of logarithmic population growth rates as a function of time.

which is just the log-likelihood per observation. Since we know and like maximum likelihood methods, why not just use this?

Well, let's think about what's going to happen if we plug in the kernel density




```
pop.cdens.o <- npcdens(popgro ~ year + factor(oecd), data = oecdpanel)
oecd.grid <- expand.grid(year = seq(from = 1965, to = 1995, by = 1), popgro = seq(from = -3.4,
    to = -2.4, length.out = 300), oecd = unique(oecdpanel$oecd))
fhat <- predict(pop.cdens.o, newdata = oecd.grid)
wireframe(fhat ~ oecd.grid$year * oecd.grid$popgro | oecd.grid$oecd, scales = list(arrows = FALSE),
    xlab = "year", ylab = "popgro", zlab = "pdf")
```

Figure 14.3 Conditional density of population growth rates given year and OECD membership. The left panel is countries not in the OECD, the right is ones which are.

estimate:

$$\frac{1}{n}\sum_{i=1}^{n}\log\left(\frac{1}{nh}\sum_{j=1}^{n}K\left(\frac{x_j - x_i}{h}\right)\right) = -\log nh + \frac{1}{n}\sum_{i=1}^{n}\log\left(\sum_{j=1}^{n}K\left(\frac{x_j - x_i}{h}\right)\right) \tag{14.49}$$

If we take h to be very small, $K(\frac{x_j - x_i}{h}) \approx 0$ unless $x_j = x_i$, so the over-all likelihood becomes

$$\approx -\log nh + \log K(0) \tag{14.50}$$

which goes to $+\infty$ as $h \to 0$. So if we want to maximize the likelihood of a kernel density estimate, we *always* want to make the bandwidth as small as possible. In fact, the limit is to say that the density is

$$\tilde{f}(x) = \frac{1}{n}\sum_{i=1}^{n} \delta(x - x_i) \tag{14.51}$$

where δ is the Dirac delta function.[9] Of course, this is just the same distribution as the empirical CDF.

Why is maximum likelihood failing us here? Well, it's doing exactly what we asked it to: to find the distribution where the observed sample is as probable as possible. Giving any probability to values of x we *didn't* see can only come at the expense of the probability of observed values, so Eq. 14.51 really is the unrestricted maximum likelihood estimate of the distribution. Anything else imposes some restrictions or constraints which don't, strictly speaking, come from the data. However, those restrictions are what let us generalize to new data, rather than just memorizing the training sample.

One way out of this is to use the *cross-validated* log-likelihood to pick a bandwidth, i.e., to restrict the sum in Eq. 14.48 to running over the testing set only. This way, very small bandwidths don't get an unfair advantage for concentrating around the training set. (If the test points are in fact all very close to the training points, then small bandwidths get a *fair* advantage.) This is in fact the default procedure in the `np` package, through the `bwmethod` option (`"cv.ml"` vs. `"cv.ls"`).

14.7 Simulating from Density Estimates

14.7.1 Simulating from Kernel Density Estimates

There are times when one wants to draw random values from the estimated distribution. This is easy with kernel density estimates, because each kernel is itself a probability density, generally a very tractable one. The pattern goes like so. Suppose the kernel is Gaussian, that we have scalar observations $x_1, x_2, \ldots x_n$, and the selected bandwidth is h. Then we pick an integer `i` uniformly at random from

[9] Recall that the delta function is defined by how it integrates with other functions: $\int \delta(x) f(x) dx = f(0)$. You can imagine $\delta(x)$ as zero everywhere except at the origin, where it has an infinitely tall, infinitely narrow spike, the area under the spike being one. If you are suspicious that this is really a bona fide function, you're right; strictly speaking it's just a linear operator on functions. We can however approximate it as the limit of well-behaved functions. For instance, take $\delta_h(x) = 1/h$ when $x \in [-h/2, h/2]$ with $\delta_h(x) = 0$ elsewhere, and let h go to zero. But this is where we came in...

1 to n, and invoke `rnorm(1,x[i],h)`.[10] Using a different kernel, we'd just need to use the random number generator function for the corresponding distribution.

To see that this gives the right distribution needs just a little math. A kernel $K(x, x_i, h)$ with bandwidth h and center x_i is a probability density function. The probability the KDE gives to any set A is just an integral:

$$\widehat{F}(A) = \int_A \widehat{f}(x)dx \tag{14.52}$$

$$= \int_A \frac{1}{n}\sum_{i=1}^{n} K(x, x_i, h)dx \tag{14.53}$$

$$= \frac{1}{n}\sum_{i=1}^{n} \int_A K(x, x_i, h)dx \tag{14.54}$$

$$= \frac{1}{n}\sum_{i=1}^{n} C(A, x_i, h) \tag{14.55}$$

introducing C to stand for the probability distribution corresponding to the kernel. The simulation procedure works if the probability that the simulated value $\tilde{X}$ falls into A matches this. To generate $\tilde{X}$, we first pick a random data point, which really means picking a random integer J, uniformly from 1 to n. Then

$$\Pr\left(\tilde{X} \in A\right) = \mathbb{E}\left[\mathbf{1}_A(\tilde{X})\right] \tag{14.56}$$

$$= \mathbb{E}\left[\mathbb{E}\left[\mathbf{1}_A(\tilde{X}) \mid J\right]\right] \tag{14.57}$$

$$= \mathbb{E}\left[C(A, x_J, h)\right] \tag{14.58}$$

$$= \frac{1}{n}\sum_{i=1}^{n} C(A, x_i, h) \tag{14.59}$$

The first step uses the fact that a probability is the expectation of an indicator function; the second uses the law of total expectation; the last steps us the definitions of C and J, and the distribution of J.

14.7.1.1 Sampling from a Joint Density

The procedure given above works with only trivial modification for sampling from a joint, multivariate distribution. If we're using a product kernel, we pick a random data point, and then draw each coordinate independently from the kernel distribution centered on our random point. (See Code Example 29 below.) The argument for correctness actually goes exactly as before.

14.7.1.2 Sampling from a Conditional Density

Sampling from a conditional density estimate with product kernels is again straightforward. The one trick is that one needs to do a *weighted* sample of data points. To see why, look at the conditional distribution (not density) function:

[10] In fact, if we want to draw a sample of size q, `rnorm(q,sample(x,q,replace=TRUE),h)` will work in R — it's important though that sampling be done *with* replacement.

$$\widehat{F}(Y \in A \mid X = x) \tag{14.60}$$

$$= \int_A \widehat{f}_{Y|X}(y \mid x)dy$$

$$= \int_A \frac{\frac{1}{nh_X h_Y}\sum_{i=1}^n K_X\left(\frac{x-x_i}{h_X}\right) K_Y\left(\frac{y-y_i}{h_Y}\right)}{\widehat{f}_X(x)}dy \tag{14.61}$$

$$= \frac{1}{nh_X h_Y \widehat{f}_X(x)} \int_A \sum_{i=1}^n K_X\left(\frac{x-x_i}{h_X}\right) K_Y\left(\frac{y-y_i}{h_Y}\right)dy \tag{14.62}$$

$$= \frac{1}{nh_X h_Y \widehat{f}_X(x)} \sum_{i=1}^n K_X\left(\frac{x-x_i}{h_X}\right) \int_A K_Y\left(\frac{y-y_i}{h_Y}\right)dy \tag{14.63}$$

$$= \frac{1}{nh_X \widehat{f}_X(x)} \sum_{i=1}^n K_X\left(\frac{x-x_i}{h_X}\right) C_Y(A, y_i, h_Y) \tag{14.64}$$

If we select the data point i with a weight proportional to $K_X\left(\frac{x-x_i}{h_X}\right)$, and then generate $\tilde{Y}$ from the K_Y distribution centered at y_i, then, $\tilde{Y}$ will follow the appropriate probability density function.

14.7.2 Drawing from Histogram Estimates

Sampling from a histogram estimate is also simple, but in a sense goes in the opposite order from kernel simulation. We first randomly pick a bin by drawing from a multinomial distribution, with weights proportional to the bin counts. Once we have a bin, we draw from a uniform distribution over its range.

14.7.3 Examples of Simulating from Kernel Density Estimates

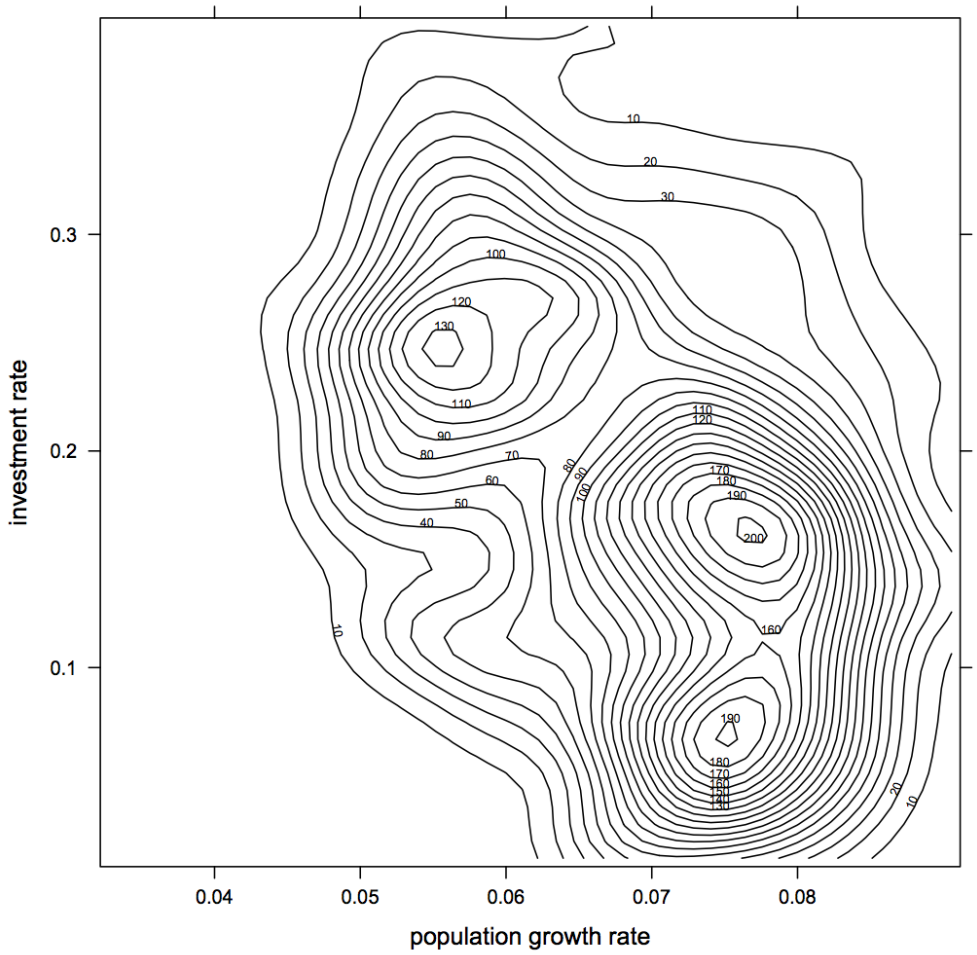
To make all this more concrete, let's continue working with the `oecdpanel` data. Section 14.4.5 shows the joint pdf estimate for the variables `popgro` and `inv` in that data set. These are the logarithms of the population growth rate and investment rate. Undoing the logarithms and taking the density gives Figure 14.4.

Let's abbreviate the actual (not logged) population growth rate as X and the actual (not logged) investment rate as Y in what follows.

Since this is a joint distribution, it implies a certain expected value for Y/X, the ratio of investment rate to population growth rate[11]. Extracting this by direct calculation from `popinv2` would not be easy; we'd need to do the integral

$$\int_{x=0}^1 \int_{y=0}^1 \frac{y}{x} \widehat{f}_{X,Y}(x, y)dydx \tag{14.65}$$

[11] Economically, we might want to know this because it would tell us about how quickly the capital stock per person grows.




```
popinv2 <- npudens(~exp(popgro) + exp(inv), data = oecdpanel)
```

Figure 14.4 Gaussian kernel density estimate for the un-logged population growth rate and investment rate. (Plotting code omitted — can you re-make the figure?)

To find $\mathbb{E}[Y/X]$ by simulation, however, we just need to generate samples from the joint distribution, say $(\tilde{X}_1, \tilde{Y}_1), (\tilde{X}_2, \tilde{Y}_2), \ldots (\tilde{X}_T, \tilde{Y}_T)$, and average:

$$\frac{1}{T}\sum_{i=1}^{T}\frac{\tilde{Y}_i}{\tilde{X}_i} = \tilde{g}_T \overset{T\to\infty}{\longrightarrow} \mathbb{E}\left[\frac{Y}{X}\right] \tag{14.66}$$

where the convergence happens because that's the law of large numbers. If the number of simulation points T is big, then $\tilde{g}_T \approx \mathbb{E}[Y/X]$. How big do we need to

```
rpopinv <- function(n) {
    n.train <- length(popinv2$dens)
    ndim <- popinv2$ndim
    points <- sample(1:n.train, size = n, replace = TRUE)
    z <- matrix(0, nrow = n, ncol = ndim)
    for (i in 1:ndim) {
        coordinates <- popinv2$eval[points, i]
        z[, i] <- rnorm(n, coordinates, popinv2$bw[i])
    }
    colnames(z) <- c("pop.growth.rate", "invest.rate")
    return(z)
}
```

CODE EXAMPLE 29: *Simulating from the fitted kernel density estimate* `popinv2`*. Can you see how to modify it to draw from other bivariate density estimates produced by* `npudens`*? From higher-dimensional distributions? Can you replace the* `for` *loop with less iterative code?*

make T? Use the central limit theorem:

$$\tilde{g}_T \rightsquigarrow \mathcal{N}(\mathbb{E}[Y/X], \mathbb{V}[\tilde{g}_1]/\sqrt{T}) \tag{14.67}$$

How do we find the variance $\mathbb{V}[\tilde{g}_1]$? We approximate it by simulating.

Code Example 29 is a function which draws from the fitted kernel density estimate. First let's check that it works, by giving it something easy to do, namely reproducing the means, which we *can* work out:

```
signif(mean(exp(oecdpanel$popgro)), 3)
## [1] 0.0693
signif(mean(exp(oecdpanel$inv)), 3)
## [1] 0.172
signif(colMeans(rpopinv(200)), 3)
## pop.growth.rate     invest.rate
##          0.0697          0.1660
```

This is pretty satisfactory for only 200 samples, so the simulator seems to be working. Now we just use it:

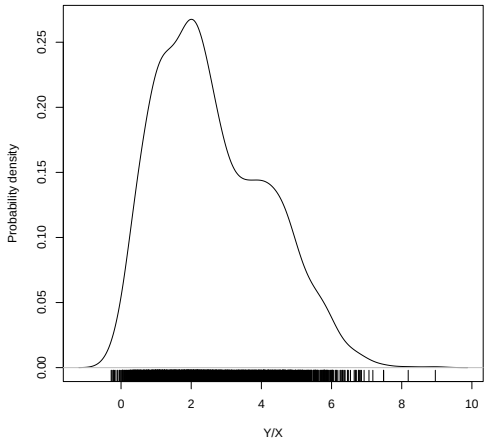
```
z <- rpopinv(2000)
signif(mean(z[, "invest.rate"]/z[, "pop.growth.rate"]), 3)
## [1] 2.61
signif(sd(z[, "invest.rate"]/z[, "pop.growth.rate"])/sqrt(2000), 3)
## [1] 0.0349
```

This tells us that $\mathbb{E}[Y/X] \approx 2.61 \pm 0.035$.

Suppose we want not the mean of Y/X but the median?

```
signif(median(z[, "invest.rate"]/z[, "pop.growth.rate"]), 3)
## [1] 2.31
```

Getting the whole distribution of Y/X is not much harder (Figure 14.5). Of course complicated things like distributions converge more slowly than simple things like means or medians, so we want might want to use more than 2000 simulated values for the distribution. Alternately, we could repeat the simulation




```
YoverX <- z[, "invest.rate"]/z[, "pop.growth.rate"]
plot(density(YoverX), xlab = "Y/X", ylab = "Probability density", main = "")
rug(YoverX, side = 1)
```

Figure 14.5 Distribution of Y/X implied by the joint density estimate `popinv2`.

many times, and look at how much variation there is from one realization to the next (Figure 14.6).

Of course, if we are going to do multiple simulations, we could just average them together. Say that $\tilde{g}_T^{(1)}$, $\tilde{g}_T^{(2)}$, $\ldots \tilde{g}_T^{(s)}$ are estimates of our statistic of interest from s independent realizations of the model, each of size T. We can just combine them into one grand average:

$$\tilde{g}_{s,T} = \frac{1}{s}\sum_{i=1}^{s} \tilde{g}_T^{(1)} \tag{14.68}$$

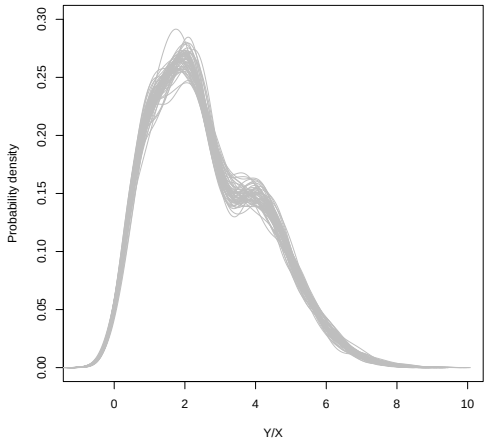
As an average of IID quantities, the variance of $\tilde{g}_{s,T}$ is $1/s$ times the variance of $\tilde{g}_T^{(1)}$.

By this point, we are getting the sampling distribution of the density of a nonlinear transformation of the variables in our model, with no more effort than calculating a mean.

14.8 Further Reading

Good introductory treatments of density estimation can be found in Simonoff (1996) and Wasserman (2006). My treatment of conditional density estimation is based on Hall *et al.* (2004).

The Glivenko-Cantelli theorem has a more "quantitative" version, the "Dvoretzky-




```
plot(0, xlab = "Y/X", ylab = "Probability density", type = "n", xlim = c(-1, 10),
    ylim = c(0, 0.3))
one.plot <- function() {
    zprime <- rpopinv(2000)
    YoverXprime <- zprime[, "invest.rate"]/zprime[, "pop.growth.rate"]
    density.prime <- density(YoverXprime)
    lines(density.prime, col = "grey")
}
invisible(replicate(50, one.plot()))
```

Figure 14.6 Showing the sampling variability in the distribution of Y/X by "over-plotting". Each line is a distribution from an estimated sample of size 2000, as in Figure 14.5; here 50 of them are plotted on top of each other. The thickness of the bands indicates how much variation there is from simulation to simulation at any given value of Y/X. (Setting the `type` of the initial `plot` to `n`, for "null", creates the plotting window, axes, legends, etc., but doesn't actually plot anything.)

Kiefer-Wolfowitz inequality", which asserts that with IID samples from a one-dimensional CDF F,

$$\Pr\left(\sup_x |\hat{F}_n(x) - F(x)| > \epsilon\right) \leq 2e^{-2n\epsilon^2} \tag{14.69}$$

and the constants appearing here are known to be the best that hold over all distributions (Wasserman, 2006, §2.2); this can be inverted to get confidence bands for the CDF.

On empirical process theory, see Pollard (1989, 1990); van de Geer (2000); Pollard (1989) is especially good as an introduction. Devroye and Lugosi (2001) applies empirical process theory to density estimation, as well as forcefully advocating measuring error using the L_1 distance, $\int |\hat{f}(x) - f(x)|dx$. In this chapter

I have stuck to L_2, partly out of tradition and partly out of desire to keep the algebra simple (which in turn helps explain the tradition).

Historical notes

I do not know of a good history of the Glivenko-Cantelli theorem (but would like to read one).

Histogram estimates are very old; the *word* "histogram" was apparently coined by Karl Pearson in the 1890s[12], but as a convenient name for an already-common type of graphic. Kernel density estimation seems to have first been proposed by Rosenblatt (1956) (see especially section 4of that paper). It was re-introduced, independently, by Parzen (1962), and some of the analysis of the error in KDEs that we saw above goes back to this paper.

Exercises

14.1 Reproduce Figure 14.4?

14.2 Qualitatively, is this compatible with Figure 14.1?

14.3 How could we use `popinv2` to calculate a joint density for `popgro` and `inv` (not `exp(popgro)` and `exp(inv)`)?

14.4 Should the density `popinv2` implies for those variables be the same as what we'd get from directly estimating their density with kernels?

14.5 You are given a kernel K which satisfies $K(u) \geq 0$, $\int K(u)du = 1$, $\int uK(u)du = 0$, $\int u^2 K(u)du = \sigma_K^2 < \infty$. You are also given a bandwidth $h > 0$, and a collection of n univariate observations $x_1, x_2, \ldots x_n$. Assume that the data are independent samples from some unknown density f.

1. Give the formula for $\hat{f}_h$, the kernel density estimate corresponding to these data, this bandwidth, and this kernel.
2. Find the expectation of a random variable whose density is $\hat{f}_h$, in terms of the sample moments, h, and the properties of the kernel function.
3. Find the variance of a random variable whose density is $\hat{f}_h$, in terms of the sample moments, h, and the properties of the kernel function.
4. How must h change as n grows to ensure that the expectation and variance of $\hat{f}_h$ will converge on the expectation and variance of f?

14.6 *The transformation method* Many variables have natural range restrictions, like being non-negative, or lie in some interval. Kernel density estimators don't respect these restrictions, so they can give positive probability density to impossible values. One way around this is the **transformation method** (or "trick"): use an invertible function q to map the limited range of X to the whole real line, find the density of the transformed variable, and then undo the transformation.

In what follows, X is a random variable with pdf f, Y is a random variable with pdf g, and $Y = q(X)$, for a known function q. You may assume that q is continuous, differentiable and monotonically increasing, inverse q^{-1} exists, and is also continuous, differentiable and monotonically increasing.

[12] See Jeff Miller (ed.), "Earliest Known Uses of Some of the Words of Mathematics", s.v. "Histogram", `http://jeff560.tripod.com/h.html`. I have not verified the references cited there, by have found the site to be generally reliable.

1. Find $g(y)$ in terms of f and q.
2. Find $f(x)$ in terms of g and q.
3. Suppose X is confined to the unit interval $[0, 1]$ and $q(x) = \log \frac{x}{1-x}$. Find $f(x)$ in terms of g and this particular q.
4. The beta distribution is confined to $[0, 1]$. Draw 1000 random values from the beta distribution with both shape parameters equal to $1/2$. Call this sample `x`, and plot its histogram. (Hint: `?rbeta`.)
5. Fit a Gaussian kernel density estimate to `x` , using `density`, `npudens`, or any other existing one-dimensional density estimator you like.
6. Find a Gaussian kernel density estimate for `logit(x)`.
7. Using your previous results, convert the KDE for `logit(x)` into a density estimate for `x` .
8. Make a plot showing (i) the true beta density, (ii) the "raw" kernel density estimate from 35, and (iii) the transformed KDE from 37. Make sure that the plotting region shows all three curves adequately, and that the three curves are visually distinct.

15

Principal Components Analysis

In Chapter 14, we saw that kernel density estimation gives us, in principle, a consistent way of nonparametrically estimating joint distributions for arbitrarily many variables. We also saw (§14.4.2) that, like regression (§8.3), density estimation suffers from the curse of dimensionality — the amount of data needed grows exponentially with the number of variables. Moreover, this is not a flaw in kernel methods, but reflects the intrinsic difficulty of the problem.

Accordingly, to go forward in multivariate data analysis, we need to somehow lift the curse of dimensionality. One approach is to hope that while we have a large number p of variables, the data is really only q-dimensional, and $q \ll p$. The next few chapters will explore various ways of finding low-dimensional structure. Alternatively, we could hope that while the data really does have lots of dimensions, it also has lots of independent parts. At an extreme, if it had p dimensions but we knew they were all statistically independent, we'd just do p one-dimensional density estimates. Chapter 18 and its sequels are concerned with this second approach, of factoring the joint distribution into independent or conditionally-independent pieces.

Principal components analysis (PCA) is one of a family of techniques for taking high-dimensional data, and using the dependencies between the variables to represent it in a more tractable, lower-dimensional form, without losing too much information. PCA is one of the simplest and most robust ways of doing such **dimensionality reduction**. The hope with PCA is that the data lie in, or close to, a low-dimensional linear subspace.

15.1 Mathematics of Principal Components

We start with p-dimensional vectors, and want to summarize them by projecting down into a q-dimensional subspace. Our summary will be the projection of the original vectors on to q directions, the **principal components**, which span the sub-space.

There are several equivalent ways of deriving the principal components mathematically. The simplest one is by finding the projections which maximize the variance. The first principal component is the direction in space along which projections have the largest variance. The second principal component is the direction which maximizes variance among all directions orthogonal to the first. The kth

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

component is the variance-maximizing direction orthogonal to the previous $k-1$ components. There are p principal components in all.

Rather than maximizing variance, it might sound more plausible to look for the projection with the smallest average (mean-squared) distance between the original vectors and their projections on to the principal components; this turns out to be equivalent to maximizing the variance (as we'll see in §15.1.1 immediately below).

Throughout, assume that the data have been "centered", so that every variable has mean 0. If we write the centered data in a matrix $\mathbf{x}$, where rows are objects and columns are variables, then $\mathbf{x}^T\mathbf{x} = n\mathbf{v}$, where $\mathbf{v}$ is the covariance matrix of the data. (You should check that last statement!)

15.1.1 Minimizing Projection Residuals

We'll start by looking for a one-dimensional projection. That is, we have p-dimensional vectors, and we want to project them on to a line through the origin. We can specify the line by a unit vector along it, $\vec{w}$, and then the projection of a data vector $\vec{x_i}$ on to the line is $\vec{x_i} \cdot \vec{w}$, which is a scalar. (Sanity check: this gives us the right answer when we project on to one of the coordinate axes.) This is the distance of the projection from the origin; the actual coordinate in p-dimensional space is $(\vec{x_i} \cdot \vec{w})\vec{w}$. The mean of the projections will be zero, because the mean of the vectors $\vec{x_i}$ is zero:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{x_i}\cdot\vec{w})\vec{w} = \left(\left(\frac{1}{n}\sum_{i=1}^{n}x_i\right)\cdot\vec{w}\right)\vec{w} \tag{15.1}$$

If we try to use our projected or **image** vectors instead of our original vectors, there will be some error, because (in general) the images do not coincide with the original vectors. (When do they coincide?) The difference is the error or **residual** of the projection. How big is it? For any one vector, say $\vec{x_i}$, it's

$$\begin{aligned}
\|\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w}\|^2 &= (\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w})\cdot(\vec{x_i} - (\vec{w}\cdot\vec{x_i})\vec{w}) && (15.2)\\
&= \vec{x_i}\cdot\vec{x_i} - \vec{x_i}\cdot(\vec{w}\cdot\vec{x_i})\vec{w} && (15.3)\\
&\quad -(\vec{w}\cdot\vec{x_i})\vec{w}\cdot\vec{x_i} + (\vec{w}\cdot\vec{x_i})\vec{w}\cdot(\vec{w}\cdot\vec{x_i})\vec{w} \\
&= \|\vec{x_i}\|^2 - 2(\vec{w}\cdot\vec{x_i})^2 + (\vec{w}\cdot\vec{x_i})^2\vec{w}\cdot\vec{w} && (15.4)\\
&= \vec{x_i}\cdot\vec{x_i} - (\vec{w}\cdot\vec{x_i})^2 && (15.5)
\end{aligned}$$

since $\vec{w}\cdot\vec{w} = \|\vec{w}\|^2 = 1$.

Add those residuals up across all the vectors:

$$\begin{aligned}
MSE(\vec{w}) &= \frac{1}{n}\sum_{i=1}^{n}\|\vec{x_i}\|^2 - (\vec{w}\cdot\vec{x_i})^2 && (15.6)\\
&= \frac{1}{n}\left(\sum_{i=1}^{n}\|\vec{x_i}\|^2 - \sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2\right) && (15.7)
\end{aligned}$$

The first summation doesn't depend on $\vec{w}$, so it doesn't matter for trying to

```
demo.theta <- runif(10, min = 0, max = pi/2)
demo.x <- cbind(cos(demo.theta), sin(demo.theta))
demo.x <- scale(demo.x, center = TRUE, scale = FALSE)
plot(demo.x, xlab = expression(x^1), ylab = expression(x^2), xlim = c(-1, 1), ylim = c(-1,
    1))
demo.w <- c(cos(-3 * pi/8), sin(-3 * pi/8))
arrows(0, 0, demo.w[1], demo.w[2], col = "blue")
text(demo.w[1], demo.w[2], pos = 4, labels = expression(w))
abline(0, b = demo.w[2]/demo.w[1], col = "blue", lty = "dashed")
projection.lengths <- demo.x %*% demo.w
projections <- projection.lengths %*% demo.w
points(projections, pch = 16, col = "blue")
segments(x0 = demo.x[, 1], y0 = demo.x[, 2], x1 = projections[, 1], y1 = projections[,
    2], col = "grey")
```

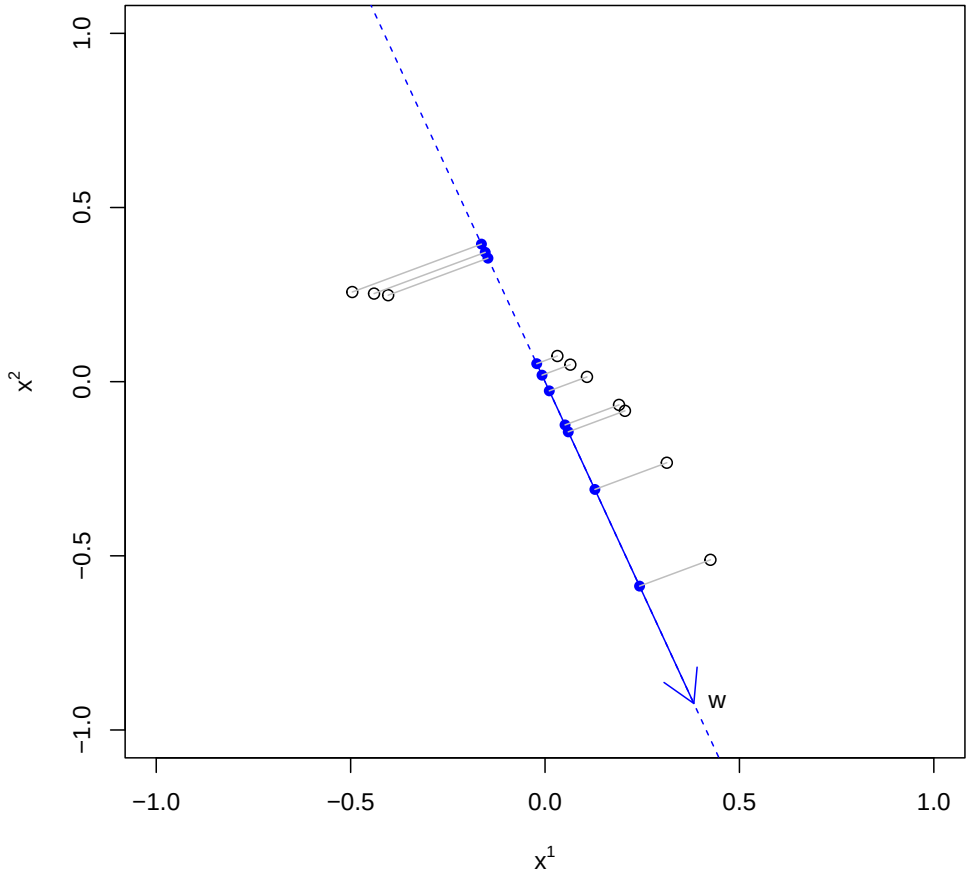


Figure 15.1 Illustration of projecting data points $\vec{x}$ (black dots) on to an arbitrary line through the space (blue, dashed), represented by a unit vector $\vec{w}$ along the line (also blue but solid). The blue dots are the projections on to the blue line, $(\vec{x} \cdot \vec{w})\vec{w}$; the gray lines are the vector residuals, $\vec{x} - (\vec{x} \cdot \vec{w})\vec{w}$. These are *not* the residuals from regressing one of the components of the data vector on the other.

minimize the mean squared residual. To make the MSE small, what we must do is make the second sum big, i.e., we want to maximize

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2 \tag{15.8}$$

which we can see is the sample mean of $(\vec{w}\cdot\vec{x_i})^2$. The (sample) mean of a square is always equal to the square of the (sample) mean plus the (sample) variance:

$$\frac{1}{n}\sum_{i=1}^{n}(\vec{w}\cdot\vec{x_i})^2 = \left(\frac{1}{n}\sum_{i=1}^{n}\vec{x_i}\cdot\vec{w}\right)^2 + \widehat{\mathbb{V}}\left[\vec{w}\cdot\vec{x_i}\right] \tag{15.9}$$

Since we've just seen that the mean of the projections is zero, minimizing the residual sum of squares is equivalent to maximizing the variance of the projections.

(Of course in general we don't want to project on to just one vector, but on to multiple principal components. If those components are orthogonal and have the unit vectors $\vec{w_1}, \vec{w_2}, \ldots \vec{w_k}$, then the image of x_i is its projection into the space spanned by these vectors,

$$\sum_{j=1}^{k}(\vec{x_i}\cdot\vec{w_j})\vec{w_j} \tag{15.10}$$

The mean of the projection on to each component is still zero. If we go through the same algebra for the mean squared error, it turns [Exercise 15.1] out that the cross-terms between different components all cancel out, and we are left with trying to maximize the sum of the variances of the projections on to the components.)

15.1.2 Maximizing Variance

Accordingly, let's maximize the variance! Writing out all the summations grows tedious, so let's do our algebra in matrix form. If we stack our n data vectors into an $n \times p$ matrix, $\mathbf{x}$, then the projections are given by $\mathbf{xw}$, which is an $n \times 1$ matrix. The variance is

$$\widehat{\mathbb{V}}\left[\vec{w}\cdot\vec{x_i}\right] = \frac{1}{n}\sum_{i}(\vec{x_i}\cdot\vec{w})^2 \tag{15.11}$$

$$= \frac{1}{n}(\mathbf{xw})^T(\mathbf{xw}) \tag{15.12}$$

$$= \frac{1}{n}\mathbf{w}^T\mathbf{x}^T\mathbf{xw} \tag{15.13}$$

$$= \mathbf{w}^T\frac{\mathbf{x}^T\mathbf{x}}{n}\mathbf{w} \tag{15.14}$$

$$= \mathbf{w}^T\mathbf{v}\mathbf{w} \tag{15.15}$$

We want to chose a unit vector $\vec{w}$ so as to maximize $\widehat{\mathbb{V}}\left[\vec{w}\cdot\vec{x_i}\right]$. To do this, we

need to make sure that we only look at unit vectors — we need to constrain the maximization. The constraint is that $\vec{w} \cdot \vec{w} = 1$, or $\mathbf{w}^T\mathbf{w} = 1$. To enforce this constraint, we introduce a Lagrange multiplier λ (Appendix D.3) and do a larger unconstrained optimization:

$$\mathcal{L}(\mathbf{w}, \lambda) \equiv \mathbf{w}^T\mathbf{v}\mathbf{w} - \lambda(\mathbf{w}^T\mathbf{w} - 1) \tag{15.16}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \mathbf{w}^T\mathbf{w} - 1 \tag{15.17}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 2\mathbf{v}\mathbf{w} - 2\lambda\mathbf{w} \tag{15.18}$$

Setting the derivatives to zero at the optimum, we get

$$\mathbf{w}^T\mathbf{w} = 1 \tag{15.19}$$

$$\mathbf{v}\mathbf{w} = \lambda\mathbf{w} \tag{15.20}$$

Thus, the desired vector $\mathbf{w}$ is an eigenvector of the covariance matrix $\mathbf{v}$, and the maximizing vector will be the one associated with the largest eigenvalue λ. This is good news, because finding eigenvectors is something which can be done comparatively rapidly, and because eigenvectors have many nice mathematical properties, which we can use as follows.

We know that $\mathbf{v}$ is a $p \times p$ matrix, so it will have at most p different eigenvectors. We know that $\mathbf{v}$ is a covariance matrix, so it is symmetric, and then linear algebra tells us that the eigenvectors must be orthogonal to one another. Again because $\mathbf{v}$ is a covariance matrix, it is a **non-negative-definite**[1] matrix, in the sense that $\vec{x} \cdot \mathbf{v}\vec{x} \geq 0$ for any $\vec{x}$. This tells us that the eigenvalues of $\mathbf{v}$ must all be ≥ 0.

The eigenvectors of $\mathbf{v}$ are the **principal components** of the data. Because we know they are orthogonal, together they span the whole p-dimensional space. The first principal component, i.e. the eigenvector which goes the largest value of λ, is the direction along which the data have the most variance. The second principal component, i.e. the second eigenvector, is the direction orthogonal to the first component with the most variance. Because it is orthogonal to the first eigenvector, their projections will be uncorrelated. In fact, projections on to all the principal components are uncorrelated with each other. If we use q principal components, our weight matrix $\mathbf{w}$ will be a $p \times q$ matrix, where each column will be a different eigenvector of the covariance matrix $\mathbf{v}$. The eigenvalues will give variance of the projection on to each component. The variance of the projections on to the first q principal components is then $\sum_{i=1}^{q} \lambda_i$.

15.1.3 More Geometry; Back to the Residuals

If we use all p principal components, the matrix $\mathbf{w}$ is a $p \times p$ matrix, where each column is an eigenvector of $\mathbf{v}$. The product $\mathbf{x}\mathbf{w}$ is a new $n \times p$ matrix, in which each column (the projection on to an eigenvector) is uncorrelated with every other column. Because eigenvectors are orthogonal and normalized, $\mathbf{w}^T\mathbf{w} = \mathbf{I}$, i.e.,

[1] Or "positive semi-definite".

$\mathbf{w}^T = \mathbf{w}^{-1}$, so $\mathbf{w}$ is itself an **orthogonal** matrix. Since the outstanding examples of orthogonal matrices are rotation matrices, $\mathbf{w}$ is often called the **rotation matrix** of the principal components analysis. It tells us how to rotate from the original coordinate system to a new system of uncorrelated coordinates.

Suppose that the data really are q-dimensional. Then $\mathbf{v}$ will have only q positive eigenvalues, and $p - q$ zero eigenvalues. If the data fall *near* a q-dimensional subspace, then $p - q$ of the eigenvalues will be nearly zero.

If we pick the top q components, we can define a projection operator $\mathbf{P}_q$. The images of the data are then $\mathbf{x}\mathbf{P}_q$. The **projection residuals** are $\mathbf{x} - \mathbf{x}\mathbf{P}_q$ or $\mathbf{x}(\mathbf{I} - \mathbf{P}_q)$. (Notice that the residuals here are vectors, not just magnitudes.) If the data really are q-dimensional, then the residuals will be zero. If the data are *approximately* q-dimensional, then the residuals will be small. In any case, we can define the R^2 of the projection as the fraction of the original variance kept by the image vectors,

$$R^2 \equiv \frac{\sum_{i=1}^{q} \lambda_i}{\sum_{j=1}^{p} \lambda_j} \tag{15.21}$$

just as the R^2 of a linear regression is the fraction of the original variance of the dependent variable kept by the fitted values.

The $q = 1$ case is especially instructive. We know that the residual vectors are all orthogonal to the projections. Suppose we ask for the first principal component of the residuals. This will be the direction of largest variance which is perpendicular to the first principal component. In other words, it will be the second principal component of the data. This suggests a recursive algorithm for finding all the principal components: the kth principal component is the leading component of the residuals after subtracting off the first $k - 1$ components. In practice, it is faster to get all the components at once from $\mathbf{v}$'s eigenvectors, but this idea is correct in principle.

This is a good place to remark that if the data really fall in a q-dimensional subspace, then $\mathbf{v}$ will have only q positive eigenvalues, because after subtracting off those components there will be no residuals. The other $p - q$ eigenvectors will all have eigenvalue 0. If the data cluster around a q-dimensional subspace, then $p - q$ of the eigenvalues will be very small, though how small they need to be before we can neglect them is a tricky question.[2]

Projections on to the first two or three principal components can be visualized; there is no guarantee, however, that only two or three dimensions really matter. Usually, to get an R^2 of 1, you need to use all p principal components.[3] How many principal components you should use depends on your data, and how big an R^2

[2] Be careful when $n < p$. Any two points define a line, and three points define a plane, etc., so if there are fewer data points than variables, it is *necessarily* true that the fall on a low-dimensional subspace. In §15.4.1, we represent stories in the New York *Times* as vectors with $p \approx 440$, but $n = 102$. Finding that only 102 principal components keep all the variance is not an empirical discovery but a mathematical artifact.

[3] The exceptions are when some of your variables are linear combinations of the others, so that you don't really have p *different* variables, or when, as just mentioned, $n < p$.

you need. Sometimes, you can get better than 80% of the variance described with just two or three components. Sometimes, however, to keep a lot of the original variance you need to use almost as many components as you had dimensions to start with.

15.1.3.1 Scree Plots

People sometimes like to make plots of the eigenvalues, in decreasing order, as in Figure 15.3. Ideally, one starts with a few big eigenvalues, and then sees a clear drop-off to a remainder of small, comparatively negligible eigenvalues. These diagrams are called **scree plots**[4]. (Some people make similar plots, but show $1 - R^2$ versus the number of components, rather than the individual eigenvalues.) Folklore recommends find the "base of the cliff" or "elbow" in the plot, the place where the number eigenvalues decrease dramatically and then level off to the right, and then retaining that number of components. This folklore appears to be based on nothing more than intuition, and offers no recommendation for what to do when there is no clear cliff or elbow in the scree plot.

15.1.4 Statistical Inference, or Not

You may have noticed, and even been troubled by, the fact that I have said nothing at all in this chapter like "assume the data are drawn at random from some distribution", or "assume the different rows of the data frame are statistically independent". This is because no such assumption is required for principal components. *All* it does is say "these data can be summarized using projections along these directions". It says nothing about the larger population or stochastic process the data came from; it doesn't even suppose there *is* a larger population or stochastic process. This is part of why §15.1.3 was so wishy-washy about the right number of components to use.

However, we could *add* a statistical assumption and see how PCA behaves under those conditions. The simplest one is to suppose that the data come iidly from a distribution with covariance matrix $\mathbf{v}_0$. Then the sample covariance matrix $\mathbf{v} \equiv n^{-1}\mathbf{x}^T\mathbf{x}$ will converge on $\mathbf{v}_0$ as $n \to \infty$. Since the principal components are smooth functions of $\mathbf{v}$ (namely its eigenvectors), they will tend to converge as n grows[5]. So, along with that additional assumption about the data-generating process, PCA does make a prediction: in the future, the principal components will look like they do now.

[4] The small loose rocks one finds at the base of cliffs or mountains are called "scree"; the metaphor is that one starts with the big eigenvalues at the top of the hill, goes down some slope, and then finds the scree beneath it, which is supposed to be negligible noise. Those who have had to cross scree fields carrying heavy camping backpacks may disagree about whether it can really be ignored.

[5] There is a wrinkle if $\mathbf{v}_0$ has "degenerate" eigenvalues, i.e., two or more eigenvectors with the same eigenvalue. Then any linear combination of those vectors is also an eigenvector, with the same eigenvalue (Exercise 15.2.) For instance, if $\mathbf{v}_0$ is the identity matrix, then every vector is an eigenvector, and PCA routines will return an essentially arbitrary collection of mutually perpendicular vectors. Generically, however, any arbitrarily small tweak to $\mathbf{v}_0$ will break the degeneracy.

Variable	Meaning
Sports	Binary indicator for being a sports car
SUV	Indicator for sports utility vehicle
Wagon	Indicator
Minivan	Indicator
Pickup	Indicator
AWD	Indicator for all-wheel drive
RWD	Indicator for rear-wheel drive
Retail	Suggested retail price (US$)
Dealer	Price to dealer (US$)
Engine	Engine size (liters)
Cylinders	Number of engine cylinders
Horsepower	Engine horsepower
CityMPG	City gas mileage
HighwayMPG	Highway gas mileage
Weight	Weight (pounds)
Wheelbase	Wheelbase (inches)
Length	Length (inches)
Width	Width (inches)

Table 15.1 *Features for the 2004 cars data.*

```
## Error in knitr(head(cars)): could not find function "knitr"
```

Table 15.2 *The first few lines of the 2004 cars data set.*

We could always add stronger statistical assumptions; in fact, Chapter 16 will look at what happens when our assumptions essentially amount to "the data lie on a low-dimensional linear subspace, plus noise". Even this, however, turns out to make PCA a not-very-attractive estimate of the statistical structure.

15.2 Example 1: Cars

Enough math; let's work an example. The data[6] consists of 388 cars from the 2004[7] model year, with 18 features. Eight features are binary indicators; the other 11 features are numerical (Table 15.1). Table 15.2 shows the first few lines from the data set. PCA only works with numerical variables, so we have ten of them to play with.

```
cars04 = read.csv("http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/data/cars-fixed04.dat")
```

There are *two* R functions for doing PCA, `princomp` and `prcomp`, which differ

[6] On the course website; from `http://www.amstat.org/publications/jse/datasets/04cars.txt`, with incomplete records removed.

[7] I realize this is a bit antiquated by the time you read this. You will finding it character-building to track down comparable data from your own time, and repeating the analysis.

in how they do the actual calculation.[8] The latter is generally more robust, so we'll just use it.

```
cars04.pca = prcomp(cars04[, 8:18], scale. = TRUE)
```

The second argument to `prcomp` tells it to first scale all the variables to have variance 1, i.e., to standardize them. You should experiment with what happens with this data when we don't standardize.

We can now extract the loadings or weight matrix from the `cars04.pca` object. For comprehensibility I'll just show the first two components.

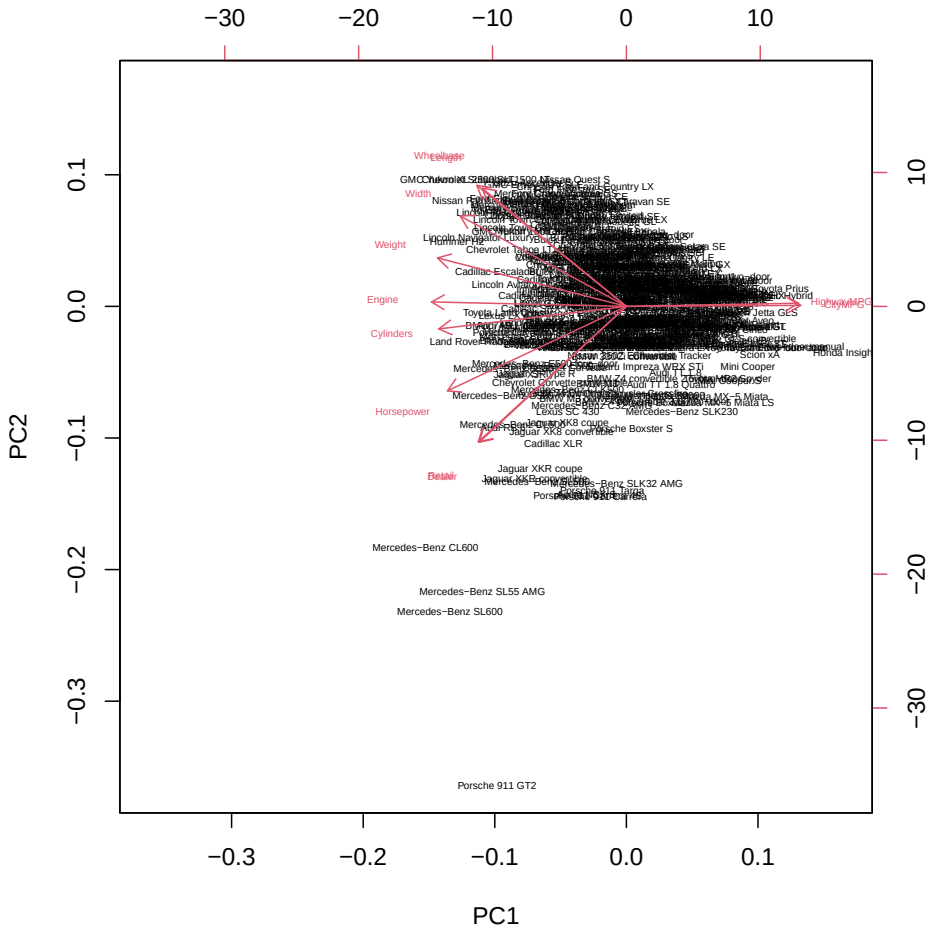
```
round(cars04.pca$rotation[, 1:2], 2)
##              PC1   PC2
## Retail     -0.26 -0.47
## Dealer     -0.26 -0.47
## Engine     -0.35  0.02
## Cylinders  -0.33 -0.08
## Horsepower -0.32 -0.29
## CityMPG     0.31  0.00
## HighwayMPG  0.31  0.01
## Weight     -0.34  0.17
## Wheelbase  -0.27  0.42
## Length     -0.26  0.41
## Width      -0.30  0.31
```

This says that all the variables *except* the gas-mileages have a negative projection on to the first component. This means that there is a negative correlation between mileage and everything else. The first principal component tells us about whether we are getting a big, expensive gas-guzzling car with a powerful engine, or whether we are getting a small, cheap, fuel-efficient car with a wimpy engine.

The second component is a little more interesting. Engine size and gas mileage hardly project on to it at all. Instead we have a contrast between the physical size of the car (positive projection) and the price and horsepower. Basically, this axis separates mini-vans, trucks and SUVs (big, not so expensive, not so much horse-power) from sports-cars (small, expensive, lots of horse-power).

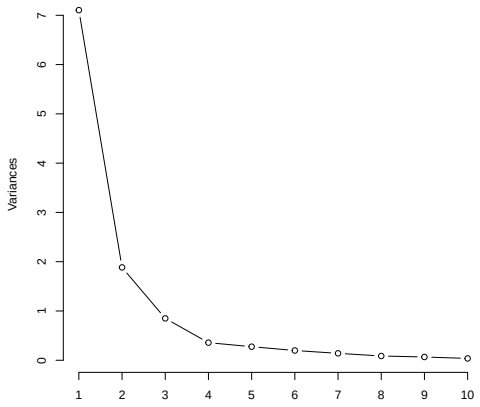
To check this interpretation, we can use a useful tool called a **biplot**, which plots the data, along with the projections of the original variables, on to the first two components (Figure 15.2). Notice that the car with the lowest value of the second component is a Porsche 911, with pick-up trucks and mini-vans at the other end of the scale. Similarly, the highest values of the first component all belong to hybrids.

[8] `princomp` actually calculates the covariance matrix and takes its eigenvalues. `prcomp` uses a different technique called "singular value decomposition".



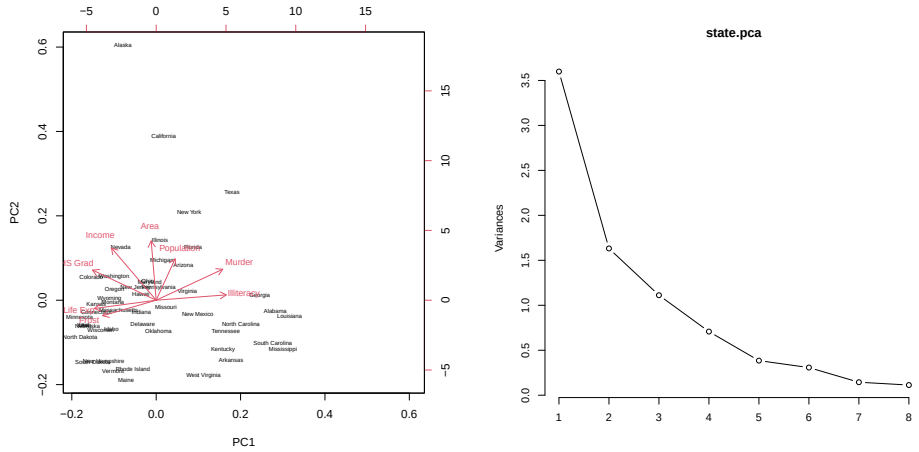

```
biplot(cars04.pca, cex = 0.4)
```

Figure 15.2 "Biplot" of the 2004 cars data. The horizontal axis shows projections on to the first principal component, the vertical axis the second component. Car names are written at their projections on to the components (using the coordinate scales on the top and the right). Red arrows show the projections of the original variables on to the principal components (using the coordinate scales on the bottom and on the left).




```
plot(cars04.pca, type = "l", main = "")
```

Figure 15.3 Scree plot of the 2004 cars data: the eigenvalues of the principal components, in decreasing order. Each eigenvalue is the variance along that component. Folklore suggests adding components until the plot levels off, or goes past an "elbow" — here this might be 2 or 3 components.




```
biplot(state.pca, cex = c(0.5, 0.75))
plot(state.pca, type = "l")
```

Figure 15.4 Biplot and scree plot for the PCA of `state.x77`.

15.3 Example 2: The United States *circa* 1977

R contains a built-in data file, `state.x77`, with facts and figures for the various states of the USA as of about 1977: population, per-capita income, the adult illiteracy rate, life expectancy, the homicide rate, the proportion of adults with at least a high-school education, the number of days of frost a year, and the state's area. While this data set is almost as old as I am[9], it still makes a convenient example, so let's step through a principal components analysis of it.

Since the variables all have different, incomparable scales, it's not a bad idea to scale them to unit variance before finding the components[10]:

```
state.pca <- prcomp(state.x77, scale. = TRUE)
```

The biplot and the scree plot (Figure 15.4) look reasonable.

With this reasonable-looking PCA, we might try to interpret the components.

```
signif(state.pca$rotation[, 1:2], 2)
##               PC1    PC2
## Population  0.130  0.410
## Income     -0.300  0.520
## Illiteracy  0.470  0.053
## Life Exp   -0.410 -0.082
## Murder      0.440  0.310
```

[9] Again, readers will find it character-building to find more modern data on which to repeat the exercise.

[10] You should try re-running all this with `scale.=FALSE`, and ponder what the experience tells you about the wisdom of advice like "maximize R^2", or even "minimize the approximation error".

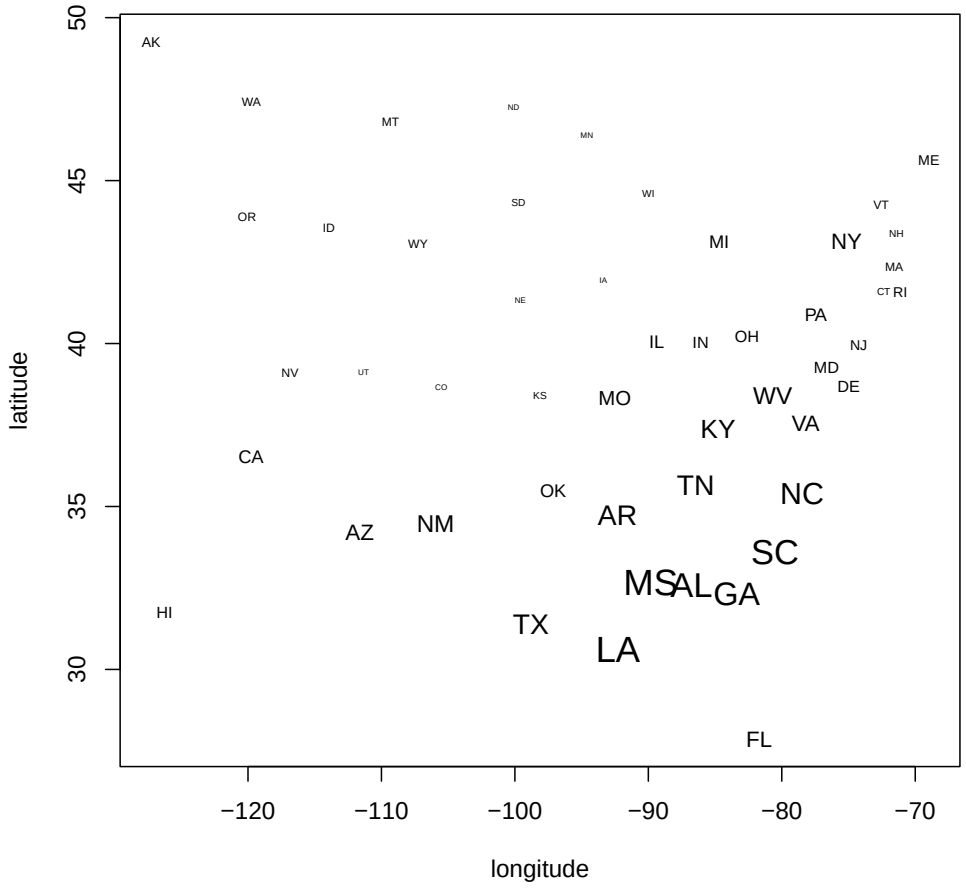
```
## HS Grad    -0.420  0.300
## Frost      -0.360 -0.150
## Area       -0.033  0.590
```

The first component aligns with illiteracy, murder, and (more weakly) population; it's negatively aligned with high school graduation, life expectancy, cold weather, income, and (very weakly) the area of the state. The second component is positively algined with area, income, population, high school graduation and murder, and negativey aligned, weakly, with cold weather and life expectancy. The first component thus separates short-lived, violent, ill-educated, poor warm states from those with the opposite qualities. The second component separates big, rich, educated, violent states from those which are small (in land or people), poor, less educated, and less violent.

Since each data point has a geographic location, we can make a map, where the sizes of the symbols for each state vary with their projection on to the first principal component. This suggests that the component is something we might call "southernness" — more precisely, the contrast between the South and the rest of the nation[11]. I will leave making a map of the second component as an exercise[12].

[11] The correlation between the first component and an indicator for being in the Confederacy is 0.8; for being a state which permitted slavery when the Civil War began, 0.78.

[12] §16.8.1 has more on this example.




```
plot.states_scaled <- function(sizes, min.size = 0.4, max.size = 2, ...) {
    plot(state.center, type = "n", ...)
    out.range = max.size - min.size
    in.range = max(sizes) - min(sizes)
    scaled.sizes = out.range * ((sizes - min(sizes))/in.range)
    text(state.center, state.abb, cex = scaled.sizes + min.size)
    invisible(scaled.sizes)
}

plot.states_scaled(state.pca$x[, 1], min.size = 0.3, max.size = 1.5, xlab = "longitude",
    ylab = "latitude")
```

Figure 15.5 The US states, plotted in their geographic locations, with symbol size varying with the projection of the state on to the first principal component. This suggests the component is something we might call "southernness".

15.4 Latent Semantic Analysis

Information retrieval systems (like search engines) and people doing computational text analysis often represent documents as what are called **bags of words**: documents are represented as vectors, where each component counts how many times each word in the dictionary appears in the text. This throws away information about word order, but gives us something we can work with mathematically. Part of the representation of one document might look like:

```
 a abandoned abc ability able about above abroad absorbed absorbing abstract
43         0   0       0    0    10     0      0        0         0        1
```

and so on through to "zebra", "zoology", "zygote", etc. to the end of the dictionary. These vectors are very, very large! At least in English and similar languages, these bag-of-word vectors have three outstanding properties:

1. Most words do not appear in most documents; the bag-of-words vectors are very **sparse** (most entries are zero).
2. A small number of words appear many times in almost all documents; these words tell us almost nothing about what the document is about. (Examples: "the", "is", "of", "for", "at", "a", "and", "here", "was", etc.)
3. Apart from those hyper-common words, most words' counts are correlated with some but not all other words; words tend to come in bunches which appear together.

Taken together, this suggests that we do not really get a lot of value from keeping around *all* the words. We would be better off if we could project down a smaller number of new variables, which we can think of as combinations of words that tend to appear together in the documents, or not at all. But this tendency needn't be absolute — it can be partial because the words mean slightly different things, or because of stylistic differences, etc. This is *exactly* what principal components analysis does.

To see how this can be useful, imagine we have a collection of documents (a **corpus**), which we want to search for documents about agriculture. It's entirely possible that many documents on this topic don't actually contain the *word* "agriculture", just closely related words like "farming". A simple search on "agriculture" will miss them. But it's very likely that the occurrence of these related words is well-correlated with the occurrence of "agriculture". This means that all these words will have similar projections on to the principal components, and it will be easy to find documents whose principal components projection is like that for a query about agriculture. This is called **latent semantic indexing**.

To see why this is *indexing*, think about what goes into coming up with an index for a book by hand. Someone draws up a list of topics and then goes through the book noting all the passages which refer to the topic, and maybe a little bit of what they say there. For example, here's the start of the entry for "Agriculture" in the index to Adam Smith's *The Wealth of Nations*:

> Agriculture, the labour of, does not admit of such subdivisions as manufactures, 6; this

impossibility of separation, prevents agriculture from improving equally with manufactures, 6; natural state of, in a new colony, 92; requires more knowledge and experience than most mechanical professions, and yet is carried on without any restrictions, 127; the terms of rent, how adjusted between landlord and tenant, 144; is extended by good roads and navigable canals, 147; under what circumstances pasture land is more valuable than arable, 149; gardening not a very gainful employment, 152–3; vines the most profitable article of culture, 154; estimates of profit from projects, very fallacious, *ib.*; cattle and tillage mutually improve each other, 220; ...

and so on. (Agriculture is an important topic in *The Wealth of Nations.*) It's asking a lot to hope for a computer to be able to do something like this, but we could at least hope for a list of pages like "6, 92, 126, 144, 147, 152–3, 154, 220,...". One could imagine doing this by treating each page as its own document, forming its bag-of-words vector, and then returning the list of pages with a non-zero entry for "agriculture". This will fail: only two of those nine pages actually contains that word, and this is pretty typical. On the other hand, they are full of words strongly correlated with "agriculture", so asking for the pages which are most similar in their principal components projection to that word will work great.[13]

At first glance, and maybe even second, this seems like a wonderful trick for extracting meaning, or **semantics**, from pure correlations. Of course there are also all sorts of ways it can fail, not least from spurious correlations. If our training corpus happens to contain lots of documents which mention "farming" and "Kansas", as well as "farming" and "agriculture", latent semantic indexing will not make a big distinction between the relationship between "agriculture" and "farming" (which is genuinely semantic, about the meaning of the words) and that between "Kansas" and "farming" (which reflects non-linguistic facts about the world, and probably wouldn't show up in, say, a corpus collected from Australia).

Despite this susceptibility to spurious correlations, latent semantic indexing is an *extremely* useful technique in practice, and the foundational papers (Deerwester *et al.*, 1990; Landauer and Dumais, 1997) are worth reading.

15.4.1 Principal Components of the New York Times

To get a more concrete sense of how latent semantic analysis works, and how it reveals semantic information, let's apply it to some data. The accompanying R file and R workspace contains some news stories taken from the New York *Times* Annotated Corpus (Sandhaus, 2008), which consists of about 1.8 million stories from the *Times*, from 1987 to 2007, which have been hand-annotated by actual human beings with standardized machine-readable information about their contents. From this corpus, I have randomly selected 57 stories about art and 45 stories about music, and turned them into a bag-of-words data frame, one row per story, one column per word; plus an indicator in the first column of whether

[13] Or it should anyway; I haven't actually done the experiment with this book.

the story is one about art or one about music.[14] The original data frame thus has 102 rows, and 4432 columns: the categorical label, and 4431 columns with counts for every distinct word that appears in at least one of the stories.[15]

The PCA is done as it would be for any other data:

[[ATTN: Why isn't loading from the course website working?]]

```
load("~/teaching/ADAfaEPoV/data/pca-examples.Rdata")
nyt.pca <- prcomp(nyt.frame[, -1])
nyt.latent.sem <- nyt.pca$rotation
```

We need to omit the first column in the first command because it contains categorical variables, and PCA doesn't apply to them. The second command just picks out the matrix of projections of the variables on to the components — this is called `rotation` because it can be thought of as rotating the coordinate axes in feature-vector space.

Now that we've done this, let's look at what the leading components are.

```
signif(sort(nyt.latent.sem[, 1], decreasing = TRUE)[1:30], 2)
##       music        trio     theater   orchestra   composers       opera
##       0.110       0.084       0.083       0.067       0.059       0.058
##    theaters           m    festival        east     program           y
##       0.055       0.054       0.051       0.049       0.048       0.048
##      jersey     players   committee      sunday        june     concert
##       0.047       0.047       0.046       0.045       0.045       0.045
##    symphony       organ     matinee   misstated instruments           p
##       0.044       0.044       0.043       0.042       0.041       0.041
##         X.d       april      samuel        jazz     pianist     society
##       0.041       0.040       0.040       0.039       0.038       0.038
signif(sort(nyt.latent.sem[, 1], decreasing = FALSE)[1:30], 2)
##       she       her        ms         i      said    mother    cooper        my
##    -0.260    -0.240    -0.200    -0.150    -0.130    -0.110    -0.100    -0.094
##  painting   process paintings        im        he       mrs        me  gagosian
##    -0.088    -0.071    -0.070    -0.068    -0.065    -0.065    -0.063    -0.062
##       was   picasso     image sculpture      baby   artists      work    photos
##    -0.058    -0.057    -0.056    -0.056    -0.055    -0.055    -0.054    -0.051
##       you    nature    studio       out      says      like
##    -0.051    -0.050    -0.050    -0.050    -0.050    -0.049
```

These are the thirty words with the largest positive and negative projections on to the first component.[16] The words with positive projections are mostly associated with music, those with negative components with the visual arts. The letters "m" and "p" show up with music because of the combination "p.m", which our parsing breaks into two single-letter words, and because stories about music give show-times more often than do stories about art. Personal pronouns appear with art stories because more of those quote people, such as artists or collectors.[17]

[14] Actually, following standard practice in language processing, I've normalized the bag-of-word vectors so that documents of different lengths are comparable, and used "inverse document-frequency weighting" to de-emphasize hyper-common words like "the" and emphasize more informative words. See the lecture notes for data mining if you're interested.

[15] If we were trying to work with the complete corpus, we should expect at least 50000 words, and perhaps more.

[16] Which direction is positive and which negative is of course arbitrary; basically it depends on internal choices in the algorithm.

[17] You should check out these explanations for yourself. The raw stories are part of the R workspace.

What about the second component?

```
signif(sort(nyt.latent.sem[, 2], decreasing = TRUE)[1:30], 2)
##         art      museum      images     artists   donations     museums
##       0.150       0.120       0.095       0.092       0.075       0.073
##    painting         tax   paintings   sculpture     gallery  sculptures
##       0.073       0.070       0.065       0.060       0.055       0.051
##     painted       white    patterns      artist      nature     service
##       0.050       0.050       0.047       0.047       0.046       0.046
##  decorative        feet     digital      statue       color    computer
##       0.043       0.043       0.043       0.042       0.042       0.041
##       paris         war collections     diamond       stone     dealers
##       0.041       0.041       0.041       0.041       0.041       0.040
signif(sort(nyt.latent.sem[, 2], decreasing = FALSE)[1:30], 2)
##          her          she      theater        opera           ms            i
##       -0.220       -0.220       -0.160       -0.130       -0.130       -0.083
##         hour   production         sang     festival        music      musical
##       -0.081       -0.075       -0.075       -0.074       -0.070       -0.070
##        songs        vocal    orchestra           la      singing      matinee
##       -0.068       -0.067       -0.067       -0.065       -0.065       -0.061
##  performance         band       awards    composers         says           my
##       -0.061       -0.060       -0.058       -0.058       -0.058       -0.056
##           im         play     broadway       singer       cooper performances
##       -0.056       -0.056       -0.055       -0.052       -0.051       -0.051
```

Here the positive words are about art, but more focused on acquiring and trading ("collections", "dealers", "donations", "dealers") than on talking with artists or about them. The negative words are musical, specifically about musical theater and vocal performances.

I could go on, but by this point you get the idea.

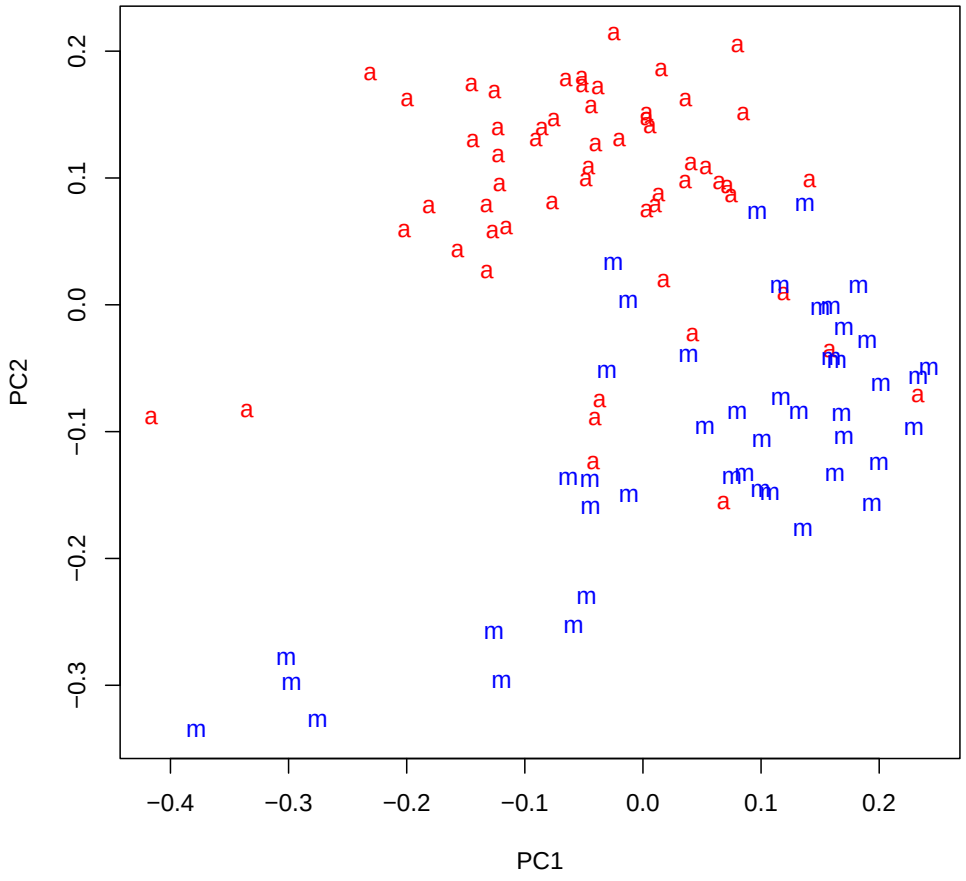
15.5 PCA for Visualization

Let's try displaying the *Times* stories using the principal components (Figure 15.6).

Notice that even though we have gone from 4431 dimensions to 2, and so thrown away a lot of information, we could draw a line across this plot and have most of the art stories on one side of it and all the music stories on the other. If we let ourselves use the first four or five principal components, we'd still have a thousand-fold savings in dimensions, but we'd be able to get almost-perfect separation between the two classes. This is a sign that PCA is really doing a good job at summarizing the information in the word-count vectors, and in turn that the bags of words give us a lot of information about the meaning of the stories.

Multidimensional scaling

Figure 15.6 also illustrates the idea of **multidimensional scaling**, which means finding low-dimensional points to represent high-dimensional data by preserving the distances between the points. If we write the original vectors as $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n$, and their images as $\vec{y}_1, \vec{y}_2, \ldots \vec{y}_n$, then the MDS problem is to pick the images to




```
plot(nyt.pca$x[, 1:2], pch = ifelse(nyt.frame[, "class.labels"] == "music", "m",
    "a"), col = ifelse(nyt.frame[, "class.labels"] == "music", "blue", "red"))
```

Figure 15.6 Projection of the *Times* stories on to the first two principal components. Music stories are marked with a blue "m", art stories with a red "a".

minimize the difference in distances:

$$\sum_{i} \sum_{j \neq i} \left(\|\vec{y}_i - \vec{y}_j\| - \|\vec{x}_i - \vec{x}_j\| \right)^2 \tag{15.22}$$

This will be small if distances between the image points are all close to the distances between the original points. PCA accomplishes this precisely because $\vec{y}_i$ is itself close to $\vec{x}_i$ (on average).

15.6 PCA Cautions

Trying to guess at what the components might mean is a good idea, but like many good ideas it's easy to go overboard. Specifically, once you attach an idea in your mind to a component, and especially once you attach a *name* to it, it's very easy to forget that those are names and ideas you made up; to **reify** them, as you might reify clusters. Sometimes the components actually do measure real variables, but sometimes they just reflect patterns of covariance which have many different causes. If I did a PCA of the same variables but for, say, European cars, I might well get a similar first component, but the second component would probably be rather different, since SUVs are much less common there than here.

A more important example comes from population genetics. Starting in the 1960s, L. L. Cavalli-Sforza and collaborators began a huge project of mapping human genetic variation — of determining the frequencies of different genes in different populations throughout the world. (Cavalli-Sforza *et al.* (1994) is the main summary; Cavalli-Sforza has also written several excellent popularizations.) For each point in space, there are a very large number of variables, which are the frequencies of the various genes among the people living there. Plotted over space, this gives a map of that gene's frequency. What they noticed (unsurprisingly) is that many genes had similar, but not identical, maps. This led them to use PCA, reducing the huge number of variables (genes) to a few components. Results look like Figure 15.7. They interpreted these components, very reasonably, as signs of large population movements. The first principal component for Europe and the Near East, for example, was supposed to show the expansion of agriculture out of the Fertile Crescent. The third, centered in steppes just north of the Caucasus, was supposed to reflect the expansion of Indo-European speakers towards the end of the Bronze Age. Similar stories were told of other components elsewhere.

Unfortunately, as Novembre and Stephens (2008) showed, spatial patterns like this are what one should expect to get when doing PCA of any kind of spatial data with local correlations, because that essentially amounts to taking a Fourier transform, and picking out the low-frequency components.[18] They simulated genetic diffusion processes, without any migration or population expansion, and got results that looked very like the real maps (Figure 15.8). This doesn't mean that the stories of the maps *must be* wrong, but it does undercut the principal components as evidence for those stories.

[18] Remember that PCA re-writes the original vectors as a weighted sum of new, orthogonal vectors, just as Fourier transforms do. When there is a lot of spatial correlation, values at nearby points are similar, so the low-frequency modes will have a lot of amplitude, i.e., carry a lot of the variance. So first principal components will tend to be similar to the low-frequency Fourier modes.

15.7 Random Projections

PCA finds the *optimal* projection from p dimensions down to q dimensions, in the sense of minimizing the MSE in Eq. 15.6. We have seen how this lead to maximizing variance, and to an elegant solution in terms of eigenvectors of the covariance matrix. However, finding eigenvectors is a lot of work. Lazy people may therefore wonder whether it would really be so bad to just use a *random* q-dimensional sub-space.

A remarkable answer is given by a geometric result which has come to be called the **Johnson-Lindenstrauss lemma**, which runs as follows. We start with n points $\vec{x}_1, \vec{x}_2, \ldots \vec{x}_n$ in $\mathbb{R}^p$, and we want to project them down to $\mathbb{R}^q$, while ensuring that all (squared) distances are preserved to within a factor of $1 \pm \epsilon$, i.e., that, for all i and j,

$$(1-\epsilon)\|\vec{x}_i - \vec{x}_j\| \leq \|\mathbf{w}\vec{x}_i - \mathbf{w}\vec{x}_j\|^2 \leq (1+\epsilon)\|\vec{x}_i - \vec{x}_j\| \tag{15.23}$$

(Compare this expression to the objective function for multidimensional scaling, Eq. 15.22.) There is always some $\mathbf{w}$ which achieves this, provided that q is at least $O(\epsilon^{-2} \log n)$. In fact, proofs of the result are constructive (Dasgupta and Gupta, 2002): we can form $\mathbf{w}$ as follows:

1. Draw $q = 4\frac{\ln n}{\epsilon/2 - \epsilon^3/3}$ uniformly-distributed, unit-length vectors[19] in $\mathbb{R}^p$;
2. Orthogonalize the vectors;
3. Scale up each vector by a factor of $\sqrt{p/q}$;
4. Make the vectors the rows of the $q \times p$ matrix $\mathbf{w}$.

The probability that this $\mathbf{w}$ keeps all of the distances between the n points to within a factor of $1 \pm \epsilon$ is at least[20] $1 - 1/n$. The fact that this probability is > 0 shows that there is *some* distance-preserving projection. Since it is easy to check whether a randomly-generated $\mathbf{w}$ does preserve the distances, we can make the probability of success as close to 1 as we like by generating multiple $\mathbf{w}$ and checking them all.

The Johnson-Lindenstrauss procedure has a number of very remarkable properties. One of them is that the projection is, indeed, completely random, and not at all a function of the data, a drastic contrast with PCA. It is plainly foolish to give any sort of interpretation to the Johnson-Lindenstrauss projection. (In fact, the *same* random projection will work with most data sets!) Another remarkable property is that the required number of dimensions q needed to approximate the data does not depend on the original dimension p. Rather, q grows, slowly, with the number of data points n. If, on the other hand, there really is a linear low-dimensional structure to the data, PCA should be able to extract it with a *fixed* number of principal components.[21]

[19] To create such a vector $\vec{U}$, make p draws Y_i from an $N(0,1)$ distribution, and set $U = (Y_1, Y_2, \ldots Y_p)/\sqrt{\sum_{i=1}^{p} Y_i^2}$.

[20] Again, see Dasgupta and Gupta (2002) for the detailed calculations; they are not too difficult, but not illuminating here.

[21] It would seem like it should be possible to turn this last point into an actual test for whether the data cluster around a linear sub-space, but, if so, I have not found where it is worked out.

15.8 Further Reading

Principal components goes back to Karl Pearson (1901). It was independently re-invented by Harold Hotelling (1933a,b), who provided the name "principal components analysis". It has been re-re-discovered many times in many fields, so it is also known as (among other things) the Karhunen-Loève transformation (see Loève (1955)), the Hotelling transformation, the method of empirical orthogonal functions, and singular value decomposition[22]. Many statistical presentations start with the idea of maximizing the variance; this seems less well-motivated to me than trying to find the best-approximating linear subspace, which was, in fact, Pearson's original goal in 1901.

The enthusiastic and well-written textbook by Eshel (2012) is largely devoted to PCA. It starts with the rudiments of linear algebra, looks into numerical and computational issues I have glossed over, and gives detailed accounts of its uses in analyzing spatial and spatio-temporal processes, especially in the Earth and environmental sciences.

As I said above, PCA is an example of a data analysis method which involves only approximation, with no statistical inference or underlying probabilistic model. Chapters 16 and 17 describe two (rather-different looking) statistical models which both imply that the data should lie in a linear subspace plus noise. Alternatively, chapter G introduces methods for approximating data with low-dimensional curved manifolds, rather than linear subspaces.

Latent semantic analysis, or latent semantic indexing, goes back to Deerwester *et al.* (1990). Hand *et al.* (2001) has a good discussion, setting it in the context of other data-analytic methods, and avoiding some of the more extravagant claims made on its behalf (Landauer and Dumais, 1997).

§15.5 just scratches the surface of the vast literature on multidimensional scaling, the general goal of which is to find low-dimensional, easily-visualized representations which are somehow faithful to the geometry of high-dimensional spaces. Much of this literature, including the name "multidimensional scaling", comes from psychology. For a brief introduction with references, I recommend Hand *et al.* (2001).

Concerns about interpretation and reification[23] are rarely very far away whenever people start using methods for finding hidden structure, whether they're just approximation methods or they attempt proper statistical inference. We will touch on them again in Chapters 16 and 17. In general, people seem to find it easier to say what's wrong or dubious about other analysts' interpretations of their components or latent constructs, than to explain what's right about their own interpretations; certainly I do.

For more on random projections, see Mahoney (2011), which sets them in

[22] Strictly speaking, singular value decomposition is a matrix algebra trick which is used in the most common algorithm for PCA.

[23] I have not been able to find out where the term "reification" comes from; some claim that it is Marxist in origin, but it's used by the early and decidedly non-Marxist Thomson (1939), so I doubt that.

the context of related randomized methods for dealing with large and/or high-dimensional data.

Exercises

15.1 Suppose that instead of projecting on to a line, we project on to a q-dimensional subspace, defined by q orthogonal length-one vectors $\vec{w_1}, \ldots \vec{w_q}$. We want to show that minimizing the mean squared error of the projection is equivalent to maximizing the sum of the variances of the scores along these q directions.

1. Write $\mathbf{w}$ for the matrix forms by stacking the $\vec{w_i}$. Prove that $\mathbf{w}^T\mathbf{w} = \mathbf{I}_q$.
2. Find the matrix of q-dimensional scores in terms of $\mathbf{x}$ and $\mathbf{w}$. *Hint:* your answer should reduce to $\vec{x_i} \cdot \vec{w_1}$ when $q = 1$.
3. Find the matrix of p-dimensional approximations based on these scores in terms of $\mathbf{x}$ and $\mathbf{w}$. *Hint:* your answer should reduce to $(\vec{x_i} \cdot \vec{w_1})\vec{w_1}$ when $q = 1$.
4. Show that the MSE of using the vectors $\vec{w_1}, \ldots \vec{w_q}$ is the sum of two terms, one of which depends only on $\mathbf{x}$ and not $\mathbf{w}$, and the other depends only on the scores along those directions (and not otherwise on what those directions are). *Hint:* look at the derivation of Eq. 15.5, and use Exercise 41.
5. Explain in what sense minimizing projection residuals is equivalent to maximizing the sum of variances along the different directions.

15.2 Suppose that $\mathbf{u}$ has two eigenvectors, $\vec{w_1}$ and $\vec{w_2}$, with the same eigenvalue a. Prove that any linear combination of $\vec{w_1}$ and $\vec{w_2}$ is also an eigenvector of $\mathbf{u}$, and also has eigenvalue a.

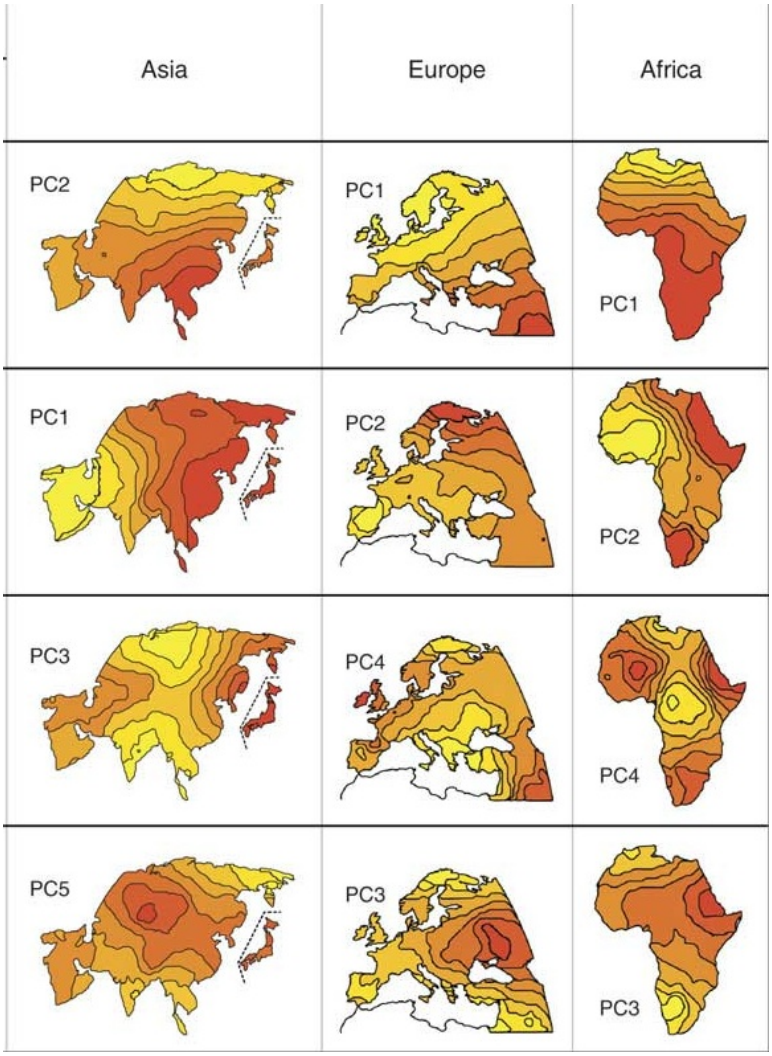


Figure 15.7 Principal components of genetic variation in the old world, according to Cavalli-Sforza *et al.* (1994), as re-drawn by Novembre and Stephens (2008).

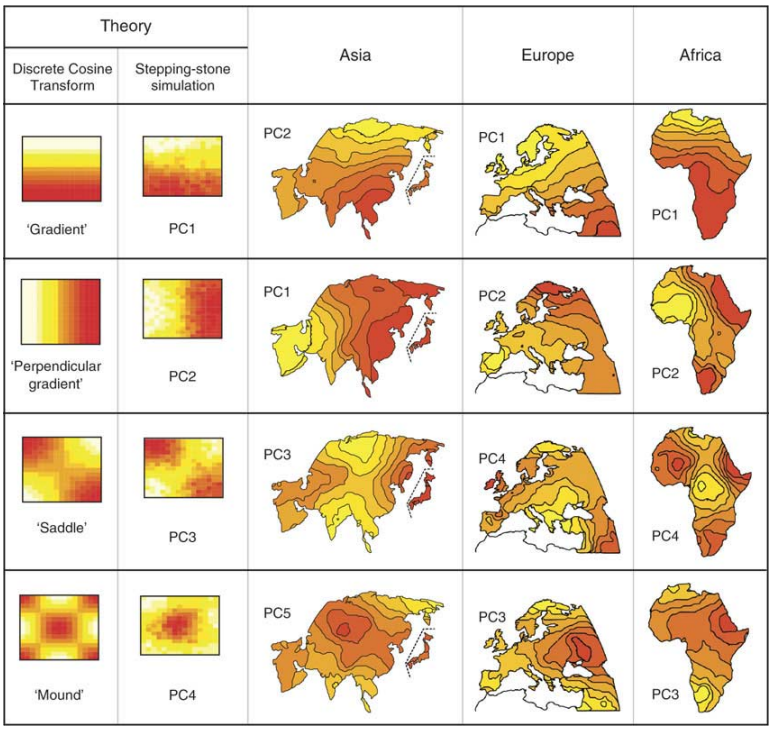


Figure 15.8 How the PCA patterns can arise as numerical artifacts (far left column) or through simple genetic diffusion (next column). From Novembre and Stephens (2008).

16

Factor Models

16.1 From PCA to Factor Models

Let's sum up PCA. We start with n different p-dimensional vectors as our data, i.e., each observation as p numerical variables. We want to reduce the number of dimensions to something more manageable, say q. The principal components of the data are the q orthogonal directions of greatest variance in the original p-dimensional space; they can be found by taking the top q eigenvectors of the sample covariance matrix. Principal components analysis summarizes the data vectors by projecting them on to the principal components. This is equivalent to finding the q-dimensional linear subspace which best approximates the data (minimizes the mean squared distance to p-dimensional vectors).

All of this is purely an algebraic undertaking; it involves no probabilistic assumptions whatsoever. It also supports no statistical inferences — saying nothing about the population or stochastic process which made the data, it just summarizes the data. How can we add some probability, and so some statistics? And what does that let us do?

Start with some notation. $\mathbf{X}$ is our data matrix, with n rows for the different observations and p columns for the different variables, so X_{ij} is the value of variable j in observation i. Each principal component is a vector of length p, and there are p of them, so we can stack them together into a $p \times p$ matrix, say $\mathbf{w}$. Finally, each data vector has a projection on to each principal component, which we collect into an $n \times p$ matrix $\mathbf{F}$. Then

$$\begin{aligned} \mathbf{X} &= \mathbf{F}\mathbf{w} \\ [n \times p] &= [n \times p][p \times p] \end{aligned} \tag{16.1}$$

where I've checked the dimensions of the matrices underneath. This is an exact equation involving no noise, approximation or error, but it's kind of useless; we've replaced p-dimensional vectors in $\mathbf{X}$ with p-dimensional vectors in $\mathbf{F}$. If we keep only to $q < p$ largest principal components, that corresponds to dropping columns from $\mathbf{F}$ and rows from $\mathbf{w}$. Let's say that the truncated matrices are $\mathbf{F}_q$ and $\mathbf{w}_q$. Then

$$\begin{aligned} \mathbf{X} &\approx \mathbf{F}_q \mathbf{w}_q \\ [n \times p] &= [n \times q][q \times p] \end{aligned} \tag{16.2}$$

The error of approximation — the difference between the left- and right- hand-

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

sides of Eq. 16.2 — will get smaller as we increase q. (The line below the equation is a sanity-check that the matrices are the right size, which they are. Also, at this point the subscript qs get too annoying, so I'll drop them.) We can of course make the two sides match exactly by adding an error or residual term on the right:

$$\mathbf{X} = \mathbf{F}\mathbf{w} + \epsilon \tag{16.3}$$

where ϵ has to be an $n \times p$ matrix.

Now, Eq. 16.3 should look more or less familiar to you from regression. On the left-hand side we have a measured outcome variable ($\mathbf{X}$), and on the right-hand side we have a systematic prediction term ($\mathbf{F}\mathbf{w}$) plus a residual (ϵ). Let's run with this analogy, and start treating ϵ as *noise*, as a random variable which has got some distribution, rather than whatever arithmetic says is needed to balance the two sides. This move is where we go from mere data reduction, making no claims about anything other than these particular data points, to actual statistical inference, making assertions about the process that generated the data. It is the difference between the difference between just drawing a straight line through a scatter plot, and inferring a linear regression.

Having made that move, $\mathbf{X}$ will also be a random variable. When we want to talk about the random variable which goes in the ith column of $\mathbf{X}$, we'll call it X_i. What about $\mathbf{F}$? Well, in the analogy it corresponds to the independent variables in the regression, which ordinarily we treat as fixed rather than random, but that's because we actually get to observe them; here we don't, so it can make sense to treat $\mathbf{F}$, too, as random. Now that they are random variables, we say that we have q **factors**, rather than components, that $\mathbf{F}$ is the matrix of **factor scores** and $\mathbf{w}$ is the matrix of **factor loadings**. The variables in $\mathbf{X}$ are called **observable** or **manifest** variables, those in $\mathbf{F}$ are **hidden** or **latent**. (Technically ϵ is also latent.)

Before we can actually do much with this model, we need to say more about the distributions of these random variables. The traditional choices are as follows.

1. All of the observable random variables X_i have mean zero and variance 1.
2. All of the latent factors have mean zero and variance 1.
3. The noise terms ϵ all have mean zero.
4. The factors are uncorrelated across individuals (rows of $\mathbf{F}$) and across variables (columns of $\mathbf{F}$).
5. The noise terms are uncorrelated across individuals, and across observable variables.
6. The noise terms are uncorrelated with the factor variables.

Item (1) isn't restrictive, because we can always center and standardize our data. Item (2) isn't restrictive either — we could always center and standardize the factor variables without really changing anything. Item (3) actually follows from (1) and (2). The substantive assumptions — the ones which will give us predictive power but could also go wrong, and so really define the factor model — are the others, about lack of correlation. Where do they come from?

Remember what the model looks like:

$$\mathbf{X} = \mathbf{F}\mathbf{w} + \epsilon \tag{16.4}$$

All of the systematic patterns in the observations $\mathbf{X}$ should come from the first term on the right-hand side. The residual term ϵ should, if the model is working, be unpredictable noise. Items (3) through (5) express a very strong form of this idea. In particular it's vital that the noise be uncorrelated with the factor scores.

16.1.1 Preserving correlations

There is another route from PCA to the factor model, which many people like but which I find less compelling; it starts by changing the objectives.

PCA aims to minimize the mean-squared distance from the data to their projects, or what comes to the same thing, to preserve variance. But it doesn't preserve correlations. That is, the correlations of the features of the image vectors are not the same as the correlations among the features of the original vectors (unless $q = p$, and we're not really doing any data reduction). We might value those correlations, however, and want to preserve them, rather than trying to approximate the actual data.[1] That is, we might ask for a set of vectors whose image in the feature space will have the same correlation matrix as the original vectors, or as close to the same correlation matrix as possible while still reducing the number of dimensions. This leads to the factor model we've already reached, as we'll see in §16.4.2.

16.2 The Graphical Model

It's common to represent factor models visually, as in Figure 16.1. This is an example of a **graphical model**, in which the nodes or vertices of the graph represent random variables, and the edges of the graph represent direct statistical dependencies between the variables. The figure shows the observables or features in square boxes, to indicate that they are manifest variables we can actual measure; above them are the factors, drawn in round bubbles to show that we don't get to see them. The fact that there are no direct linkages between the factors shows that they are independent of one another. From below we have the noise terms, one to an observable.

Notice that not every observable is connected to every factor: this depicts the fact that some entries in $\mathbf{w}$ are zero. In the figure, for instance, X_1 has an arrow only from F_1 and not the other factors; this means that while $w_{11} = 0.87$, $w_{21} = w_{31} = 0$.

[1] Why? Well, originally the answer was that the correlation coefficient had just been invented, and was about the only way people had of measuring relationships between variables. Since then it's been propagated by statistics courses where it is the only way people are *taught* to measure relationships. The great statistician John Tukey once wrote "Does anyone know when the correlation coefficient is useful, as opposed to when it is used? If so, why not tell us?" (Tukey, 1954, p. 721).

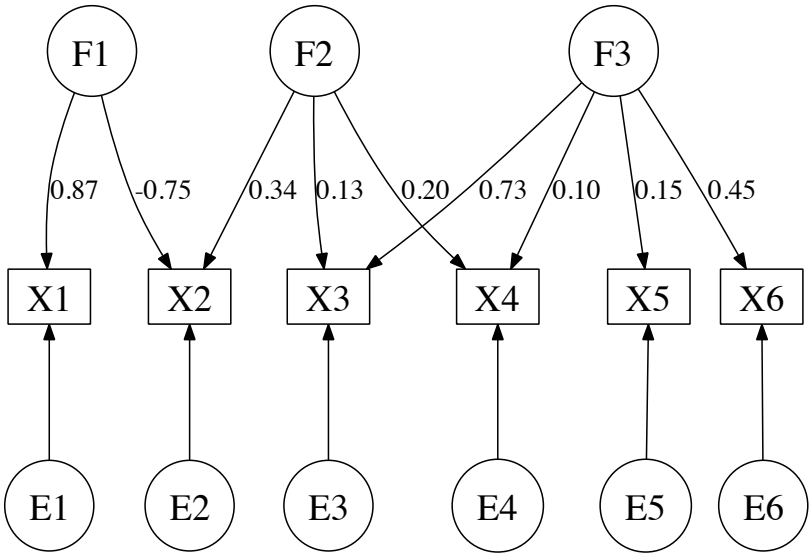


Figure 16.1 Graphical model form of a factor model. Circles stand for the unobserved variables (factors above, noises below), boxes for the observed features. Edges indicate non-zero coefficients — entries in the factor loading matrix $\mathbf{w}$, or specific variances ψ_i. Arrows representing entries in $\mathbf{w}$ are decorated with those entries. Note that it is common to omit the noise variables in such diagrams, with the implicit understanding that every variable with an incoming arrow also has an incoming noise term.

Drawn this way, one sees how the factor model is **generative** — how it gives us a recipe for producing new data. In this case, it's:

- draw new, independent values for the factor scores $F_1, F_2, \ldots F_q$;
- add these up with weights from $\mathbf{w}$; and then
- add on the final noises $\epsilon_1, \epsilon_2, \ldots \epsilon_p$.

If the model is right, this is a procedure for generating new, synthetic data with the same characteristics as the real data. In fact, it's a story about how the real data came to be — that there really are some latent variables (the factor scores) which linearly cause the observables to have the values they do.

16.2.1 Observables Are Correlated Through the Factors

One of the most important consequences of the factor model is that observable variables are correlated with each other *solely* because they are correlated with the hidden factors. To see how this works, take X_1 and X_2 from the diagram, and let's calculate their covariance. (Since they both have variance 1, this is the same as their correlation.)

$$\begin{aligned}
\mathrm{Cov}\left[X_1, X_2\right] &= \mathbb{E}\left[X_1 X_2\right] - \mathbb{E}\left[X_1\right]\mathbb{E}\left[X_2\right] && (16.5)\\
&= \mathbb{E}\left[X_1 X_2\right] && (16.6)\\
&= \mathbb{E}\left[(F_1 w_{11} + F_2 w_{21} + \epsilon_1)(F_1 w_{12} + F_2 w_{22} + \epsilon_2)\right] && (16.7)\\
&= \mathbb{E}\left[F_1^2 w_{11} w_{12} + F_1 F_2 (w_{11} w_{22} + w_{21} w_{12}) + F_2^2 w_{21} w_{22}\right] \\
&\quad + \mathbb{E}\left[\epsilon_1 \epsilon_2\right] + \mathbb{E}\left[\epsilon_1 (F_1 w_{12} + F_2 w_{22})\right] \\
&\quad + \mathbb{E}\left[\epsilon_2 (F_1 w_{11} + F_2 w_{21})\right] && (16.8)
\end{aligned}$$

Since the noise terms are uncorrelated with the factor scores, and the noise terms for different variables are uncorrelated with each other, *all* the terms containing ϵs have expectation zero. Also, F_1 and F_2 are uncorrelated, so

$$\begin{aligned}
\mathrm{Cov}\left[X_1, X_2\right] &= \mathbb{E}\left[F_1^2\right] w_{11} w_{12} + \mathbb{E}\left[F_2^2\right] w_{21} w_{22} && (16.9)\\
&= w_{11} w_{12} + w_{21} w_{22} && (16.10)
\end{aligned}$$

using the fact that the factors are scaled to have variance 1. This says that the covariance between X_1 and X_2 is what they have from both correlating with F_1, plus what they have from both correlating with F_2; if we had more factors we would add on $w_{31} w_{32} + w_{41} w_{42} + \ldots$ out to $w_{q1} w_{q2}$. And of course this would apply as well to any other pair of observable variables. So the general form is

$$\mathrm{Cov}\left[X_i, X_j\right] = \sum_{k=1}^{q} w_{ki} w_{kj} \tag{16.11}$$

so long as $i \neq j$.

The jargon says that observable i **loads on** factor k when $w_{ki} \neq 0$. If two observables do not load on to any of the same factors, if they do not share any common factors, then they will be independent. If we could condition on ("control for") the factors, all of the observables would be conditionally independent.

Graphically, we draw an arrow from a factor node to an observable node if and only if the observable loads on the factor. So then we can just *see* that two observables are correlated if they both have in-coming arrows from the same factors. (To find the actual correlation, we multiply the weights on all the edges connecting the two observable nodes to the common factors; that's Eq. 16.11.) Conversely, even though the factors are marginally independent of each other, if two factors both send arrows to the same observable, then they are dependent conditional on that observable.[2]

[2] To see that this makes sense, suppose that $X_1 = F_1 w_{11} + F_2 w_{21} + \epsilon_1$. If we know the value of X_1, we know what F_1, F_2 and ϵ_1 have to add up to, so they are conditionally dependent.

16.2.2 Geometry: Approximation by Linear Subspaces

Each observation we take is a vector in a p-dimensional space; the factor model says that these vectors have certain geometric relations to each other — that the data has a certain shape. To see what that is, pretend for right now that we can turn off the noise terms ϵ. The loading matrix $\mathbf{w}$ is a $q \times p$ matrix, so each row of $\mathbf{w}$ is a vector in p-dimensional space; call these vectors $\vec{w}_1, \vec{w}_2, \ldots \vec{w}_q$. Without the noise, our observable vectors would be linear combinations of these vectors (with the factor scores saying how much each vector contributes to the combination). Since the factors are orthogonal to each other, we know that they span a q-dimensional subspace of the p-dimensional space — a line if $q = 1$, a plane if $q = 2$, in general a linear subspace. If the factor model is true and we turn off noise, we would find all the data lying *exactly* on this subspace. Of course, *with* noise we expect that the data vectors will be scattered around the subspace; how close depends on the variance of the noise. (Figure 16.2.) But this is still a rather specific prediction about the shape of the data.

A weaker prediction than "the data lie on a low-dimensional linear subspace in the high-dimensional space" is "the data lie on some low-dimensional surface, possibly curved, in the high-dimensional space"; there are techniques for trying to recover such surfaces. Chapter G introduces two such techniques, but this is a broad and still-growing area.

16.3 Roots of Factor Analysis in Causal Discovery

The roots of factor analysis go back to work by Charles Spearman just over a century ago (Spearman, 1904); he was trying to discover the hidden structure of human intelligence. His observation was that schoolchildren's grades in different subjects were all correlated with each other. He went beyond this to observe a particular *pattern* of correlations, which he thought he could explain as follows: the reason grades in math, English, history, etc., are all correlated is performance in these subjects is all correlated with *something else*, a general or **common** factor, which he named "general intelligence", for which the natural symbol was of course g or G.

Put in a form like Eq. 16.4, Spearman's model becomes

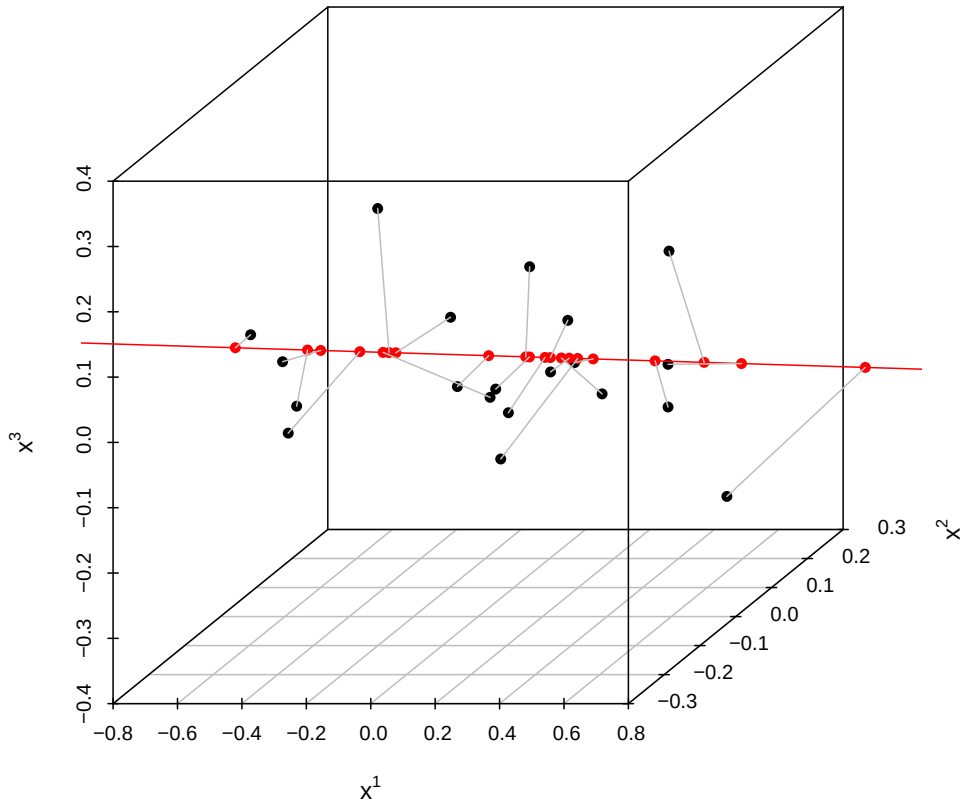
$$\mathbf{X} = \epsilon + \mathbf{G}\mathbf{w} \tag{16.12}$$

where $\mathbf{G}$ is an $n \times 1$ matrix (i.e., a row vector) and $\mathbf{w}$ is a $1 \times p$ matrix (i.e., a column vector). The correlation between feature i and G is just $w_i \equiv w_{1i}$, and, if $i \neq j$,

$$\mathbf{v}_{ij} \equiv \mathrm{Cov}\left[X_i, X_j\right] = w_i w_j \tag{16.13}$$

where I have introduced $\mathbf{v}_{ij}$ as a short-hand for the covariance.

Up to this point, this is all so much positing and assertion and hypothesis. What Spearman did next, though, was to observe that this hypothesis carried a very strong implication about the *ratios* of correlation coefficients. Pick any four




```
n <- 20; library(scatterplot3d)
f <- matrix(sort(rnorm(n)),ncol=1); w <- matrix(c(0.5,0.2,-0.1),nrow=1)
fw <- f %*% w; x <- fw + matrix(rnorm(n*3,sd=c(.15,.05,.09)),ncol=3,byrow=TRUE)
s3d <- scatterplot3d(x,xlab=expression(x^1),ylab=expression(x^2),
                     zlab=expression(x^3),pch=16)
s3d$points3d(matrix(seq(from=min(f)-1,to=max(f)+1,length.out=2),ncol=1)%*%w,
             col="red",type="l")
s3d$points3d(fw,col="red",pch=16)
for (i in 1:nrow(x)) {
  s3d$points3d(x=c(x[i,1],fw[i,1]),y=c(x[i,2],fw[i,2]),z=c(x[i,3],fw[i,3]),
               col="grey",type="l") }
```

Figure 16.2 Geometry of factor models: Black dots are observed vectors $\vec{X}$ in $p = 3$ dimensions. These were generated from the $q = 1$ dimensional factor scores $\vec{F}$ by taking $\vec{F}\mathbf{w}$ (red dots) and adding independent noise (grey lines). The q-dimensional subspace along which all values of $\vec{F}\mathbf{w}$ must fall is also shown in red. (See also exercise 16.3.)

distinct features, i, j, k, l. Then, if the model (16.12) is true,

$$\frac{\mathbf{v}_{ij}/\mathbf{v}_{kj}}{\mathbf{v}_{il}/\mathbf{v}_{kl}} = \frac{w_i w_j / w_k w_j}{w_i w_l / w_k w_l} \tag{16.14}$$

$$= \frac{w_i/w_k}{w_i/w_k} \tag{16.15}$$

$$= 1 \tag{16.16}$$

The relationship

$$\mathbf{v}_{ij}\mathbf{v}_{kl} = \mathbf{v}_{il}\mathbf{v}_{kj} \tag{16.17}$$

is called the "tetrad equation", and we will meet it again later when we consider methods for causal discovery in Part III. In Spearman's model, this is one tetrad equation for every set of four distinct variables.

Spearman found that the tetrad equations held in his data on school grades (to a good approximation), and concluded that a single general factor of intelligence must exist[3]. This was, of course, logically fallacious.

Later work, using large batteries of different kinds of intelligence tests, showed that the tetrad equations do not hold in general, or more exactly that departures from them are too big to explain away as sampling noise. (Recall that the equations are about the true correlations between the variables, but we only get to see sample correlations, which are always a little off.) The response, done in an *ad hoc* way by Spearman and his followers, and then more systematically by Thurstone, was to introduce *multiple* factors. This breaks the tetrad equation, but still accounts for the correlations among features by saying that features are really directly correlated with factors, and uncorrelated conditional on the factor scores. Thurstone's form of factor analysis is basically the one people still use — there have been refinements, of course, but it's mostly still his method.

16.4 Estimation

The factor model introduces a whole bunch of new variables to explain the observables: the factor scores $\mathbf{F}$, the factor loadings or weights $\mathbf{w}$, and the observable-specific variances ψ_i. The factor scores are specific to each individual, and individuals by assumption are independent, so we can't expect them to really generalize. But the loadings $\mathbf{w}$ and the variances ψ are, supposedly, characteristic of the population. So it would be nice if we could separate estimating the population parameters from estimating the attributes of individuals; here's how.

Since the variables are centered, we can write the covariance matrix in terms of the data frames:

$$\mathbf{v} = \mathbb{E}\left[\frac{1}{n}\mathbf{X}^T\mathbf{X}\right] \tag{16.18}$$

[3] Actually, the equations didn't hold when music was one of the grades, so Spearman argued musical ability did not load on general intelligence.

(This is the true, population covariance matrix on the left.) But the factor model tells us that

$$\mathbf{X} = \mathbf{F}\mathbf{w} + \epsilon \tag{16.19}$$

This involves the factor scores $\mathbf{F}$, but remember that when we looked at the correlations between individual variables, those went away, so let's substitute Eq. 16.19 into Eq. 16.18 and see what happens:

$$\mathbb{E}\left[\frac{1}{n}\mathbf{X}^T\mathbf{X}\right] \tag{16.20}$$

$$= \frac{1}{n}\mathbb{E}\left[(\epsilon^T + \mathbf{w}^T\mathbf{F}^T)(\mathbf{F}\mathbf{w} + \epsilon)\right] \tag{16.21}$$

$$= \frac{1}{n}\left(\mathbb{E}\left[\epsilon^T\epsilon\right] + \mathbf{w}^T\mathbb{E}\left[\mathbf{F}^T\epsilon\right] + \mathbb{E}\left[\epsilon^T\mathbf{F}\right]\mathbf{w} + \mathbf{w}^T\mathbb{E}\left[\mathbf{F}^T\mathbf{F}\right]\mathbf{w}\right) \tag{16.22}$$

$$= \psi + 0 + 0 + \frac{1}{n}\mathbf{w}^T n\mathbf{I}\mathbf{w} \tag{16.23}$$

$$= \psi + \mathbf{w}^T\mathbf{w} \tag{16.24}$$

Behold:

$$\mathbf{v} = \psi + \mathbf{w}^T\mathbf{w} \tag{16.25}$$

The individual-specific variables $\mathbf{F}$ have gone away, leaving only population parameters on both sides of the equation. The left-hand side is clearly something we can learn reliably from data, so if we can solve this equation for ψ and $\mathbf{w}$, we can estimate the factor parameter's models. Can we solve Eq. 16.25?

16.4.1 Degrees of Freedom

It only takes a bit of playing with Eq. 16.25 to realize that we are in trouble. Like any matrix equation, it represents a system of equations. How many equations in how many unknowns? Naively, we'd say that we have p^2 equations (one for each element of the matrix $\mathbf{v}$), and $p + pq$ unknowns (one for each diagonal element of ψ, plus one for each element of $\mathbf{w}$). If there are more equations than unknowns, then there is generally no solution; if there are fewer equations than unknowns, then there are generally infinitely many solutions. Either way, solving for $\mathbf{w}$ seems hopeless (unless $q = p - 1$, in which case it's not very helpful). What to do?

Well, first let's do the book-keeping for degrees of freedom more carefully. The observables variables are scaled to have standard deviation one, so the diagonal entries of $\mathbf{v}$ are all 1. Moreover, any covariance matrix is symmetric, so we are left with only $p(p-1)/2$ degrees of freedom in $\mathbf{v}$; there are really only that many equations. On the other side, scaling to standard deviation 1 means we don't *really* need to solve separately for ψ, because it's fixed as soon as we know what $\mathbf{w}^T\mathbf{w}$ is, which saves us p unknowns. Also, the entries in $\mathbf{w}$ are not completely free to vary independently of each other, because each row has to be orthogonal to every other row. (Look back at Chapter 15.) Since there are q rows, this gives

us $q(q-1)/2$ constraints on $\mathbf{w}$ — we can think of these as either extra equations, or as reductions in the number of free parameters (unknowns).[4]

Summarizing, we really have $p(p-1)/2$ degrees of freedom in $\mathbf{v}$, and $pq - q(q-1)/2$ degrees of freedom in $\mathbf{w}$. If these two match, then there is (in general) a unique solution which will give us $\mathbf{w}$. But in general they will *not* be equal; then what? Let us consider the two cases.

More unknowns (free parameters) than equations (constraints)

This is fairly straightforward: there is no unique solution to Eq. 16.25; instead there are *infinitely many* solutions. It's true that the loading matrix $\mathbf{w}$ does have to satisfy some constraints, that not just any $\mathbf{w}$ will work, so the data does give us some information, but there is a continuum of different parameter settings which are all match the covariance matrix perfectly. (Notice that we are working with the population parameters here, so this isn't an issue of having only a limited sample.) There is just no way to use data to decide between these different parameters, to identify which one is right, so we say the model is **unidentifiable**. Most software for factor analysis, include R's `factanal` function, will check for this and just refuse to fit a model with too many factors relative to the number of observables.

More equations (constraints) than unknowns (free parameters)

This is more interesting. In general, systems of equations like this are **overdetermined**, meaning that there is no way to satisfy all the constraints at once, and there are generally no solutions. We just can't get all possible covariance matrices $\mathbf{v}$ among, say, $p = 7$ variables in terms of, say, $q = 1$ factor models (as $p(p-1)/2 = 21$ but $pq - q(q-1)/2 = 7$). But it *is* possible for special covariance matrices. In these situations, the factor model actually has testable implications for the data — it says that only certain covariance matrices are possible and not others. For example, we saw above that the one-fator model implies the tetrad equations must hold among the observable covariances; the constraints on $\mathbf{v}$ for multiple-factor models are similar in kind but more complicated algebraically. By testing these implications, we can check whether or not our favorite factor model is right.[5]

Now we don't know the true, population covariance matrix $\mathbf{v}$, but we can estimate it from data, getting an estimate $\widehat{\mathbf{v}}$. The natural thing to do then is to equate this with the parameters and try to solve for the latter:

$$\widehat{\mathbf{v}} = \widehat{\psi} + \widehat{\mathbf{w}}^T \widehat{\mathbf{w}} \tag{16.26}$$

The book-keeping for degrees of freedom here is the same as for Eq. 16.25. If q is

[4] Notice that $\psi + \mathbf{w}^T\mathbf{w}$ is automatically symmetric, since ψ is diagonal, so we don't need to impose any extra constraints to get symmetry.

[5] We need to be a little careful here. If we find that the tetrad equations don't hold, we know a one-factor model must be wrong. We could only conclude that the one-factor model must be right if we found that the tetrad equations held, *and* that there were no other models which implied those equations; but, as we'll see, there are (§16.9.

too large relative to p, the model is unidentifiable; if it is too small, the matrix equation can only be solved if $\widehat{\mathbf{v}}$ is of the right, restricted form, i.e., if the model is right. Of course even if the model is right, the sample covariances are the true covariances plus noise, so we shouldn't expect to get an *exact* match, but we can try in various way to minimize the discrepancy between the two sides of the equation.

16.4.2 A Clue from Spearman's One-Factor Model

Remember that in Spearman's model with a single general factor, the covariance between observables i and j in that model is the product of their factor weightings:

$$\mathbf{v}_{ij} = w_i w_j \tag{16.27}$$

The exception is that $\mathbf{v}_{ii} = w_i^2 + \psi_i$, rather than w_i^2. However, if we look at $\mathbf{u} = \mathbf{v} - \psi$, that's the same as $\mathbf{v}$ off the diagonal, and a little algebra shows that its diagonal entries are, in fact, just w_i^2. So if we look at any two rows of $\mathbf{U}$, they're proportional to each other:

$$\mathbf{u}_{ij} = \frac{w_i}{w_k}\mathbf{u}_{kj} \tag{16.28}$$

This means that, when Spearman's model holds true, there is actually only *one* linearly-independent row in in $\mathbf{u}$.

Recall from linear algebra that the **rank** of a matrix is how many linearly independent rows it has.[6] Ordinarily, the matrix is of **full rank**, meaning all the rows are linearly independent. What we have just seen is that when Spearman's model holds, the matrix $\mathbf{u}$ is *not* of full rank, but rather of rank 1. More generally, when the factor model holds with q factors, the matrix $\mathbf{u} = \mathbf{w}^T\mathbf{w}$ has rank q. The diagonal entries of $\mathbf{u}$, called the **common variances** or **commonalities**, are no longer automatically 1, but rather show how much of the variance in each observable is associated with the variances of the latent factors. Like $\mathbf{v}$, $\mathbf{u}$ is a positive symmetric matrix.

Because $\mathbf{u}$ is a positive symmetric matrix, we know from linear algebra that it can be written as

$$\mathbf{u} = \mathbf{c}\mathbf{d}\mathbf{c}^T \tag{16.29}$$

where $\mathbf{c}$ is the matrix whose columns are the eigenvectors of $\mathbf{u}$, and $\mathbf{d}$ is the diagonal matrix whose entries are the eigenvalues. That is, if we use all p eigenvectors, we can reproduce the covariance matrix exactly. Suppose we instead use $\mathbf{c_q}$, the $p \times q$ matrix whose columns are the eigenvectors going with the q largest eigenvalues, and likewise make $\mathbf{d_q}$ the diagonal matrix of those eigenvalues. Then $\mathbf{c_q}\mathbf{d_q}\mathbf{c_q}^T$ will be a symmetric positive $p \times p$ matrix. This is a matrix of rank q, and so can only equal $\mathbf{u}$ if the latter also has rank q. Otherwise, it's an approximation which grows more accurate as we let q grow towards p, and, at any given q, it's a

[6] We could also talk about the columns; it wouldn't make any difference.

better approximation to $\mathbf{u}$ than any other rank-q matrix. This, finally, is the precise sense in which factor analysis tries to preserve correlations: $\mathbf{u}$ just contains information about the correlations, and we're going to try to approximate $\mathbf{u}$ as well as possible.

To resume our algebra, define $\mathbf{d_q}^{1/2}$ as the $q \times q$ diagonal matrix of the square roots of the eigenvalues. Clearly $\mathbf{d_q} = \mathbf{d_q}^{1/2}\mathbf{d_q}^{1/2}$. So

$$\mathbf{c_q}\mathbf{d_q}\mathbf{c_q}^T = \mathbf{c_q}\mathbf{d_q}^{1/2}\mathbf{d_q}^{1/2}\mathbf{c_q}^T = \left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)\left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)^T \tag{16.30}$$

So we have

$$\mathbf{u} \approx \left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)\left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)^T \tag{16.31}$$

but at the same time we know that $\mathbf{u} = \mathbf{w}^T\mathbf{w}$. So we just identify $\mathbf{w}$ with $\left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)^T$:

$$\mathbf{w} = \left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)^T \tag{16.32}$$

and we are done with our algebra.

Let's think a bit more about how well we're approximating $\mathbf{v}$. The approximation will always be exact when $q = p$, so that there is one factor for each feature (in which case $\psi = 0$ always). Then all factor analysis does for us is to rotate the coordinate axes in feature space, so that the new coordinates are uncorrelated. (This is the same as what PCA does with p components.) The approximation can *also* be exact with fewer factors than features if the reduced covariance matrix is of less than full rank, and we use at least as many factors as the rank.

16.4.3 Estimating Factor Loadings and Specific Variances

The classical method for estimating the factor model is now simply to do this eigenvector approximation on the *sample* correlation matrix. Define the **reduced** or **adjusted** sample correlation matrix as

$$\widehat{\mathbf{u}} = \widehat{\mathbf{v}} - \widehat{\psi} \tag{16.33}$$

We can't actually calculate $\widehat{\mathbf{u}}$ until we know, or have a guess as to, $\widehat{\psi}$. A reasonable and common starting-point is to do a linear regression of each feature j on all the other features, and then set $\widehat{\psi_j}$ to the mean squared error for that regression. (We'll come back to this guess later.)

Once we have the reduced correlation matrix, find its top q eigenvalues and eigenvectors, getting matrices $\widehat{\mathbf{c_q}}$ and $\widehat{\mathbf{d_q}}$ as above. Set the factor loadings accord-

ingly, and re-calculate the specific variances:

$$\widehat{\mathbf{w}} = \left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)^T \tag{16.34}$$

$$\widehat{\psi_j} = 1 - \sum_{r=1}^{q} w_{rj}^2 \tag{16.35}$$

$$\tilde{\mathbf{v}} \equiv \widehat{\psi} + \widehat{\mathbf{w}}^T\widehat{\mathbf{w}} \tag{16.36}$$

The "predicted" covariance matrix $\tilde{\mathbf{v}}$ in the last line is exactly right on the diagonal (by construction), and should be closer off-diagonal than anything else we could do with the same number of factors. However, our guess as to $\mathbf{u}$ depended on our initial guess about ψ, which has in general changed, so we can try iterating this (i.e., re-calculating $\mathbf{c_q}$ and $\mathbf{d_q}$), until we converge.

16.4.4 Maximum Likelihood Estimation

It has probably not escaped your notice that the estimation procedure above requires a starting guess as to ψ. This makes its consistency somewhat shaky. (If we continually put in ridiculous values for ψ, why should we expect that $\widehat{\mathbf{w}} \to \mathbf{w}$?) On the other hand, we know from our elementary statistics courses that maximum likelihood estimates are generally consistent, unless we choose a spectacularly bad model. Can we use that here?

We can, but at a cost. We have so far got away with just making assumptions about the means and covariances of the factor scores $\mathbf{F}$. To get an actual likelihood, we need to assume something about their distribution as well.

The usual assumption is that $F_{ik} \sim \mathcal{N}(0,1)$, and that the factor scores are independent across factors $k = 1, \ldots q$ and individuals $i = 1, \ldots n$. With this assumption, the features have a multivariate normal distribution $\vec{X}_i \sim \mathcal{N}(0, \psi + \mathbf{w}^T\mathbf{w})$. This means that the log-likelihood is

$$L = -\frac{np}{2}\log 2\pi - \frac{n}{2}\log|\psi + \mathbf{w}^T\mathbf{w}| - \frac{n}{2}\,\mathrm{tr}\left((\psi + \mathbf{w}^T\mathbf{w})^{-1}\widehat{\mathbf{v}}\right) \tag{16.37}$$

where $\mathrm{tr}\,\mathbf{a}$ is the **trace** of the matrix $\mathbf{a}$, the sum of its diagonal elements. Notice that the likelihood only involves the data through the sample covariance matrix $\widehat{\mathbf{v}}$ — the actual factor scores $\mathbf{F}$ are not needed for the likelihood.

One can either try direct numerical maximization, or use a two-stage procedure. Starting, once again, with a guess as to ψ, one finds that the crucial quantity is actually $\psi^{1/2}\mathbf{w}^T$, the optimal value of which is given by the matrix whose columns are the q leading eigenvectors of $\psi^{1/2}\widehat{\mathbf{v}}\psi^{1/2}$. Starting from a guess as to $\mathbf{w}$, the optimal choice of ψ is given by the diagonal entries of $\widehat{\mathbf{v}} - \mathbf{w}^T\mathbf{w}$. So again one starts with a guess about the unique variances (e.g., the residuals of the regressions) and iterates to convergence.[7]

The differences between the maximum likelihood estimates and the "principal

[7] The algebra is tedious. See section 3.2 in Bartholomew (1987) if you really want it. (Note that Bartholomew has a sign error in his equation 3.16.)

factors" approach can be substantial. If the data appear to be normally distributed (as shown by the usual tests), then the additional efficiency of maximum likelihood estimation is highly worthwhile. Also, as we'll see below, it is a lot easier to test the model assumptions if one uses the MLE.

16.4.5 Alternative Approaches

Factor analysis is an example of trying to approximate a full-rank matrix, here the covariance matrix, with a low-rank matrix, or a low-rank matrix plus some corrections, here $\psi + \mathbf{w}^T\mathbf{w}$. Such matrix-approximation problems are currently the subject of very intense interest in statistics and machine learning, with many new methods being proposed and refined, and it is very plausible that some of these will prove to work better than older approaches to factor analysis.

In particular, Kao and Van Roy (2013) have used these ideas to propose a new factor-analysis algorithm, which simultaneously estimates the number of factors and the factor loadings, and does so through a modification of PCA, distinct from the old "principal factors" method. In their examples, it works better than conventional approaches, but whether this will hold true generally is not clear. They do not, unfortunately, provide code.

16.4.6 Estimating Factor Scores

Given $\mathbf{X}$ and (estimates of) the parameters, it's natural to want to estimate the factor scores $\mathbf{F}$. One of the best methods for doing so is the "regression" or "Thomson" method, which says

$$\widehat{F}_{ir} = \sum_j X_{ij} b_{jr} \tag{16.38}$$

and seeks the weights b_{jr} which will minimize the mean squared error, $\mathbb{E}\left[(\widehat{F}_{ir} - F_{ir})^2\right]$. You can work out the b_{jr} as an exercise (16.6), assuming you know $\mathbf{w}$ and ψ.

16.5 The Rotation Problem

Recall from linear algebra that a matrix $\mathbf{o}$ is **orthogonal** if its inverse is the same as its transpose, $\mathbf{o}^T\mathbf{o} = \mathbf{I}$. The classic examples are rotation matrices. For instance, to rotate a two-dimensional vector through an angle α, we multiply it by

$$\mathbf{r}_\alpha = \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} \tag{16.39}$$

The inverse to this matrix must be the one which rotates through the angle $-\alpha$, $\mathbf{r}_\alpha^{-1} = \mathbf{r}_{-\alpha}$, but trigonometry tells us that $\mathbf{r}_{-\alpha} = \mathbf{r}_\alpha^T$.

To see why this matters to us, go back to the matrix form of the factor model,

and insert an orthogonal $q \times q$ matrix and its transpose:

$$\mathbf{X} = \epsilon + \mathbf{F}\mathbf{w} \tag{16.40}$$
$$= \epsilon + \mathbf{F}\mathbf{o}\mathbf{o}^T\mathbf{w} \tag{16.41}$$
$$= \epsilon + \mathbf{H}\mathbf{y} \tag{16.42}$$

We've changed the factor scores to $\mathbf{H} \equiv \mathbf{F}\mathbf{o}$, and we've changed the factor loadings to $\mathbf{y} \equiv \mathbf{o}^T\mathbf{w}$, but nothing about the features has changed *at all*. We can do as many orthogonal transformations of the factors as we like, with no observable consequences whatsoever.[8]

Statistically, the fact that different parameter settings give us the same observational consequences means that the parameters of the factor model are **unidentifiable**. The rotation problem is, as it were, the revenant of having an ill-posed problem: we thought we'd slain it through heroic feats of linear algebra, but it's still around and determined to have its revenge.[9]

Mathematically, this should not be surprising at all. The factors live in a q-dimensional vector space of their own. We should be free to set up any coordinate system we feel like on that space. Changing coordinates in factor space will just require a compensating change in how factor-space coordinates relate to feature space (the factor loadings matrix $\mathbf{w}$). That's all we've done here with our orthogonal transformation.

Substantively, this should be rather troubling. If we can rotate the factors as much as we like without consequences, how on Earth can we interpret them?

16.6 Factor Analysis as a Predictive Model

Unlike principal components analysis, factor analysis really does give us a predictive model. Its prediction is that if we draw a new member of the population and look at the vector of observables we get from them,

$$\vec{X} \sim \mathcal{N}(0, \mathbf{w}^T\mathbf{w} + \psi) \tag{16.43}$$

if we make the usual distributional assumptions. Of course it might seem like it makes a more refined, conditional prediction,

$$\vec{X}|\vec{F} \sim \mathcal{N}(F\mathbf{w}, \psi) \tag{16.44}$$

[8] Notice that the log-likelihood only involves $\mathbf{w}^T\mathbf{w}$, which is equal to $\mathbf{w}^T\mathbf{o}\mathbf{o}^T\mathbf{w} = \mathbf{y}^T\mathbf{y}$, so even assuming Gaussian distributions doesn't let us tell the difference between the original and the transformed variables. In fact, if $\vec{F} \sim \mathcal{N}(0, \mathbf{I})$, then $\vec{F}\mathbf{o} \sim \mathcal{N}(0\mathbf{o}, \mathbf{o}^T\mathbf{I}\mathbf{o}) = \mathcal{N}(0, \mathbf{I})$ — in other words, the rotated factor scores still satisfy our distributional assumptions.

[9] Remember that we obtained the loading matrix $\mathbf{w}$ as a solution to $\mathbf{w}^T\mathbf{w} = \mathbf{u}$, that is we got $\mathbf{w}$ as a kind of matrix square root of the reduced correlation matrix. For a real number u there are two square roots, i.e., two numbers w such that $w \times w = u$, namely the usual $w = \sqrt{u}$ and $w = -\sqrt{u}$, because $(-1) \times (-1) = 1$. Similarly, whenever we find one solution to $\mathbf{w}^T\mathbf{w} = \mathbf{u}$, $\mathbf{o}^T\mathbf{w}$ is another solution, because $\mathbf{o}\mathbf{o}^T = \mathbf{I}$. So while the usual "square root" of $\mathbf{u}$ is $\mathbf{w} = \mathbf{d_q}^{1/2}\mathbf{c}$, for any orthogonal matrix $\mathbf{o}^T\mathbf{d_q}^{1/2}\mathbf{c}$ will always work just as well.

but the problem is that there is no way to guess at or estimate the factor scores $\vec{F}$ until after we've seen $\vec{X}$, at which point anyone can predict X perfectly. So the actual *forecast* is given by Eq. 16.43.[10]

Now, without going through the trouble of factor analysis, one could always just postulate that

$$\vec{X} \sim \mathcal{N}(0, \mathbf{v}) \tag{16.45}$$

and estimate $\mathbf{v}$; the maximum likelihood estimate of it is the observed covariance matrix, but really we could use any consistent estimator. The closer ours is to the true $\mathbf{v}$, the better our predictions. One way to think of factor analysis is that it looks for the maximum likelihood estimate, but constrained to matrices of the form $\mathbf{w}^T\mathbf{w} + \psi$.

On the plus side, the constrained estimate has a faster rate of convergence. That is, both the constrained and unconstrained estimates are consistent and will converge on their optimal, population values as we feed in more and more data, but for the same amount of data the constrained estimate is probably closer to its limiting value. In other words, the constrained estimate $\widehat{\mathbf{w}}^T\widehat{\mathbf{w}} + \widehat{\psi}$ has less variance than the unconstrained estimate $\widehat{\mathbf{v}}$.

On the minus side, maybe the true, population $\mathbf{v}$ just can't be written in the form $\mathbf{w}^T\mathbf{w} + \psi$. Then we're getting biased estimates of the covariance and the bias will *not* go away, even with infinitely many samples. Using factor analysis rather than just fitting a multivariate Gaussian means betting that either this bias is really zero, or that, with the amount of data on hand, the reduction in variance outweighs the bias.

(I haven't talked about estimation errors in the parameters of a factor model. With large samples and maximum-likelihood estimation, one could use the usual asymptotic theory. For small samples, one bootstraps as usual.)

16.6.1 How Many Factors?

How many factors should we use? All the tricks people use for the how-many-principal-components question can be tried here, too, with the obvious modifications. However, some other answers can also be given, using the fact that the factor model does make predictions, unlike PCA.

1. *Likelihood ratio tests* Sample covariances will almost never be exactly equal to population covariances. So even if the data comes from a model with q factors, we can't expect the tetrad equations (or their multi-factor analogs) to hold exactly. The question then becomes whether the observed covariances are compatible with sampling fluctuations in a q-factor model, or are too big for that.

[10] A subtlety is that we might get to see some but not all of $\vec{X}$, and use that to predict the rest. Say $\vec{X} = (X_1, X_2)$, and we see X_1. Then we could, in principle, compute the conditional distribution of the factors, $p(F|X_1)$, and use that to predict X_2. Of course one could do the same thing using the correlation matrix, factor model or no factor model.

We can tackle this question by using log likelihood ratio tests. The crucial observations are that a model with q factors is a special case of a model with $q+1$ factors (just set a row of the weight matrix to zero), and that in the most general case, $q = p$, we can get *any* covariance matrix $\mathbf{v}$ into the form $\mathbf{w}^T\mathbf{w}$. (Set $\psi = 0$ and proceed as in the "principal factors" estimation method.)

For the usual asymptotic-theory reasons (App. [[REF]]), $\widehat{\theta}$ is the maximum likelihood estimate in a restricted model with s parameters, and $\widehat{\Theta}$ is the MLE in a more general model with $r > s$ parameters, containing the former as a special case, and finally ℓ is the log-likelihood function

$$2[\ell(\widehat{\Theta}) - \ell(\widehat{\theta})] \rightsquigarrow \chi^2_{r-s} \tag{16.46}$$

when the data came from the small model. The general regularity conditions needed for this to hold apply to Gaussian factor models, so we can test whether one factor is enough, two, etc.

(Said another way, adding another factor never reduces the likelihood, but the equation tells us how much to *expect* the log-likelihood to go up when the new factor really adds nothing and is just over-fitting the noise.)

Determining q by getting the smallest one without a significant result in a likelihood ratio test is fairly traditional, but statistically messy.[11] To raise a subject we'll return to, if the true $q > 1$ and all goes well, we'll be doing lots of hypothesis tests, and making sure this compound procedure works reliably is harder than controlling any one test. Perhaps more worrisomely, calculating the likelihood relies on distributional assumptions for the factor scores and the noises, which are hard to check for latent variables.

2. If you are comfortable with the distributional assumptions, use Eq. 16.43 to predict new data, and see which q gives the best predictions — for comparability, the predictions should be compared in terms of the log-likelihood they assign to the testing data. If genuinely new data is not available, use cross-validation.

   Comparative prediction, and especially cross-validation, seems to be somewhat rare with factor analysis, for no good reason.

16.6.1.1 R^2 and Goodness of Fit

For PCA, we saw that R^2 depends on the sum of the eigenvalues 15.1.3. For factor models, the natural notion of R^2 is the sum of squared factor loadings:

$$R^2 = \frac{\sum_{j=1}^{q}\sum_{k=1}^{p} w_{jk}^2}{p} \tag{16.47}$$

(Remember that the factors are, by design, uncorrelated with each other, and that the entries of $\mathbf{w}$ are the correlations between factors and observables.) If we write $\mathbf{w}$ in terms of eigenvalues and eigenvectors as in §16.4.2, $\mathbf{w} = \left(\mathbf{c_q}\mathbf{d_q}^{1/2}\right)^T$, then you can show that the numerator in R^2 is, again, a sum of eigenvalues.

[11] Suppose q is really 1, but by chance that gets rejected. Whether $q = 2$ gets rejected in turn is not an independent event!

People sometimes select the number of factors by looking at how much variance they "explain" — really, how much variance is kept after smoothing on to the plane. As usual with model selection by R^2, there is little good to be said for this, except that it is fast and simple.

In particular, R^2 should not be used to assess the goodness-of-fit of a factor model. The bluntest way to see this is to simulate data which does *not* come from a factor model, fit a small number of factors, and see what R^2 one gets. This was done by Peterson (2000), who found that it was easy to get R^2 of 0.4 or 0.5, and sometimes even higher[12] The same paper surveyed values of R^2 from the published literature on factor models, and found that the typical value was also somewhere around 0.5; no doubt this was just a coincidence[13].

Instead of looking at R^2, it is much better to check goodness-of-fit by actually goodness-of-fit tests. In the particular case of factor models with the Gaussian assumption, we can use a log-likelihood ratio test, checking the null hypothesis that the number of factors $= q$ against the alternative of an arbitrary multivariate Gaussian (which is the same as p factors). This test is automatically performed by `factanal` in R.

If the Gaussian assumption is dubious but we want a factor model and goodness-of-fit anyway, we can look at the difference between the empirical covariance matrix $\mathbf{v}$ and the one estimated by the factor model, $\widehat{\psi} + \widehat{\mathbf{w}}^T\widehat{\mathbf{w}}$. There are several notions of distance between matrices (matrix norms) which could be used as test statistics; a simple one is to use the sum of squared differences between the entries of $\mathbf{v}$ and those of $\widehat{\psi} + \widehat{\mathbf{w}}^T\widehat{\mathbf{w}}$. (This is the square of the "Frobenius" norm of $\mathbf{v} - (\widehat{\psi} + \widehat{\mathbf{w}}^T\widehat{\mathbf{w}})$.) Sampling distributions would have to come from bootstrapping, where we would want to simulate from the factor model.

16.7 Factor Models versus PCA Once More

We began this chapter by seeking to add some noise, and some probabilistic assumptions, into PCA. The factor models we came up with are closely related to principal components, but are *not* the same. Many of the differences have been mentioned as we went, but it's worth collecting some of the most important ones here.

1. Factor models assume that the data comes from a certain distribution, IID across data points. PCA assumes nothing about distributions at all. Moreover, factor models can be used *generatively*, to say how the latent factors cause the observable variables. PCA has nothing to say about the data-generating process.
2. Factor models can be tested by their predictions on new data points; PCA cannot.

[12] See also `http://bactra.org/weblog/523.html` for a similar experiment, with (not very elegant) R code.

[13] Peterson (2000) also claims that reported values of R^2 for PCA are roughly equal to those of factor analysis, but by this point I hope that none of you take that as an argument in favor of PCA.

3. Factor models assume that the variance matrix of the data takes a special form, $\mathbf{w}^T\mathbf{w} + \psi$, where $\mathbf{w}$ is $q \times p$ and ψ is diagonal. That is, the variance matrix must be "low rank plus noise". PCA works no matter what sample variance matrix the data might have.
4. If the factor model *is* true, then the principal components are (or approach with enough data) the eigenvectors of $\mathbf{w}^T\mathbf{w} + \psi$. They do not approach the eigenvectors of $\mathbf{w}^T\mathbf{w}$, which would be the principal factors. If the noise is small, the difference may also be small, but the factor model can be correct, if perhaps not so useful, while ψ is as big as $\mathbf{w}^T\mathbf{w}$.
5. Factor models are subject to the rotation problem; PCA is not. Which one has the advantage here is unclear.
6. Similarly, a principal component is just a linear combination of the observable variables. A latent factor is another, distinct random variable. Differences in factor scores imply differences in the *expected* values of observables. Differences in projections on to principal components imply differences in realized values of observables. (It's a little like the distinction between the predicted value for the response in a linear regression, which is a combination of the covariates, and the actual value of the response.)

16.8 Examples in R

16.8.1 Example 1: Back to the US circa 1977

We resume looking at the properties of the US states around 1977. In §15.3, we did a principal components analysis, finding a first component that seemed to mark the distinction between the South and the rest of the country, and a second that seemed to separate big, rich states from smaller, poorer ones. Let's now subject the data to factor analysis. We begin with one factor, using the base R function `factanal`.

```
(state.fa1 <- factanal(state.x77,factors=1,scores="regression"))
##
## Call:
## factanal(x = state.x77, factors = 1, scores = "regression")
##
## Uniquenesses:
## Population     Income Illiteracy   Life Exp     Murder    HS Grad      Frost
##      0.957      0.791      0.235      0.437      0.308      0.496      0.600
##       Area
##      0.998
##
## Loadings:
##            Factor1
## Population -0.208
## Income      0.458
## Illiteracy -0.875
## Life Exp    0.750
## Murder     -0.832
## HS Grad     0.710
```

```
## Frost       0.632
## Area
##
##                Factor1
## SS loadings      3.178
## Proportion Var   0.397
##
## Test of the hypothesis that 1 factor is sufficient.
## The chi square statistic is 91.97 on 20 degrees of freedom.
## The p-value is 3.34e-11
```

The output here tells us what fraction of the variance in each observable comes from its own noise (= the diagonal entries in $\widehat{\psi}$ = "uniquenesses"). It also gives us the factor loadings, i.e., the rows of $\widehat{\mathbf{w}}$. Here there's only one loading vector, since we set `factors` $= q = 1$. As a courtesy, the default printing method for the loadings leaves blanks where the loadings would be very small (here, for `Area`); this can be controlled through options (see `help(loadings)`). The last option picks between different methods of estimating the factor scores.

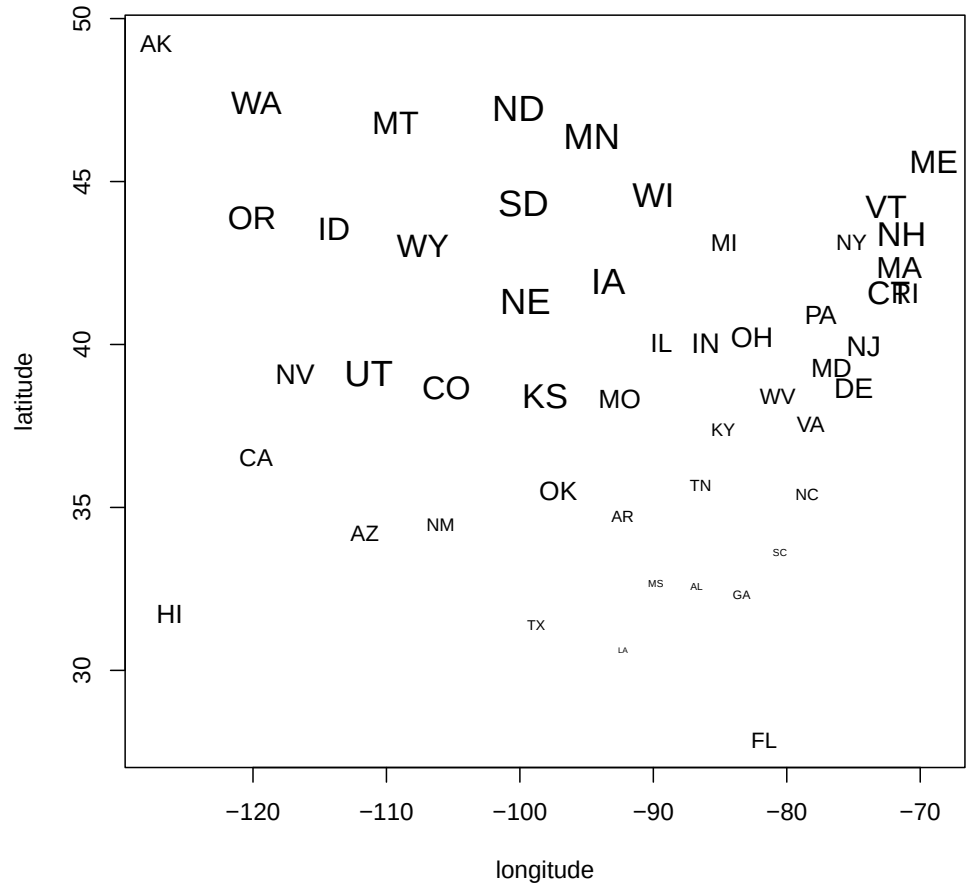
For comparison, here is the first principal component:

```
##
## Loadings:
##            PC1    PC2    PC3    PC4    PC5    PC6    PC7    PC8
## Population  0.126  0.411 -0.656 -0.409  0.406                -0.219
## Income     -0.299  0.519 -0.100        -0.638  0.462
## Illiteracy  0.468                0.353         0.387 -0.620 -0.339
## Life Exp   -0.412        -0.360  0.443  0.327  0.219 -0.256  0.527
## Murder      0.444  0.307  0.108 -0.166 -0.128 -0.325 -0.295  0.678
## HS Grad    -0.425  0.299         0.232        -0.645 -0.393 -0.307
## Frost      -0.357 -0.154  0.387 -0.619  0.217  0.213 -0.472
## Area                0.588  0.510  0.201  0.499  0.148  0.286
##
##                  PC1   PC2   PC3   PC4   PC5   PC6   PC7   PC8
## SS loadings    1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
## Proportion Var 0.125 0.125 0.125 0.125 0.125 0.125 0.125 0.125
## Cumulative Var 0.125 0.250 0.375 0.500 0.625 0.750 0.875 1.000
```

The first principal component is clearly not the same as the single common factor we extracted, even after a sign change, but it's not shockingly dissimilar either, as a map shows.

Of course, why use just one factor? Given the number of observables, we can fit up to four factors before the problem becomes totally unidentified and `factanal` refuses to work. That function automatically runs the likelihood ratio test every time it fits a model, assuming Gaussian distributions for the observables. As remarked (p. 387, this can work reasonably well for other distributions if they're not too non-Gaussian, especially if n is much larger than the number of parameters; of course $n = 50$ is pretty modest. Still, let's try it:

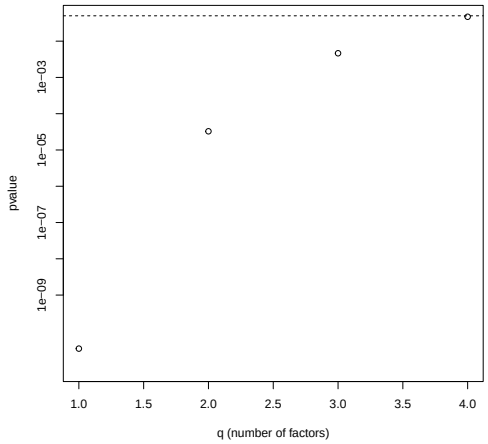
```
pvalues <- sapply(1:4,function(q){factanal(state.x77,factors=q)$PVAL})
signif(pvalues,2)
## objective objective objective objective
##   3.3e-11   3.3e-05   4.6e-03   4.7e-02
```




```
plot.states_scaled(state.fa1$score[,1],min.size=0.3,max.size=1.5,
                   xlab="longitude",ylab="latitude")
```

Figure 16.3 The US states, plotted in position with symbols scaled by their factor scores in a one-factor model. Compare to Figure 15.5, which is where the `plot.states_scaled` function comes from. (Try plotting the negative of the factor scores to make the maps look more similar.)

(Figure 16.4 plots the results.) None of the models has a p-value crossing the conventional 0.05 level, meaning all of them show systematic, detectable departures from what the data should look like if the factor model were true. Still, the four-factor model comes close.




```
plot(1:4,pvalues,xlab="q (number of factors)", ylab="pvalue",
     log="y",ylim=c(1e-11,0.04))
abline(h=0.05,lty="dashed")
```

Figure 16.4 Gaussian likelihood ratio test p-value for models with various numbers of latent factors, fit to the US-in-1977 data.

Notice that the first factor's loadings do not stay the same when we add more factors, unlike the first principal component:

```
print(factanal(state.x77, factors=4)$loadings)
##
## Loadings:
##            Factor1 Factor2 Factor3 Factor4
## Population                          0.636
## Income      0.313   0.281   0.561   0.189
## Illiteracy -0.466  -0.878
## Life Exp    0.891   0.191
## Murder     -0.792  -0.384   0.109   0.405
## HS Grad     0.517   0.418   0.581
## Frost       0.128   0.679   0.105  -0.460
## Area       -0.174           0.796
##
##                Factor1 Factor2 Factor3 Factor4
## SS loadings      2.054   1.680   1.321   0.821
## Proportion Var   0.257   0.210   0.165   0.103
## Cumulative Var   0.257   0.467   0.632   0.734
```

16.8.2 Example 2: Stocks

Classical financial theory suggests that the log-returns of corporate stocks should be IID Gaussian random variables, but allows for the possibility that different

stocks might be correlated with each other. In fact, theory suggests that the returns to any given stock should be the sum of two components: one which is specific to that firm, and one which is common to all firms. (More specifically, the common component is one which couldn't be eliminated even in a perfectly diversified portfolio.) This in turn implies that stock returns should match a one-factor model. Further investigation of this idea is deferred to Data-Analysis Assigment 8.

16.9 Reification, and Alternatives to Factor Models

A natural impulse, when looking at something like Figure 16.1, is to **reify** the factors, and to treat the arrows **causally**: that is, to say that there really is *some* variable corresponding to each factor, and that changing the value of that variable will change the features. For instance, one might want to say that there is a real, physical variable corresponding to the factor F_1, and that increasing this by one standard deviation will, on average, increase X_1 by 0.87 standard deviations, decrease X_2 by 0.75 standard deviations, and do nothing to the other features. Moreover, changing any of the other factors has no effect on X_1.

Sometimes all this is even right. How can we tell when it's right?

16.9.1 The Rotation Problem Again

Consider the following matrix, call it $\mathbf{r}$:

$$\begin{bmatrix} \cos 30 & -\sin 30 & 0 \\ \sin 30 & \cos 30 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{16.48}$$

Applied to a three-dimensional vector, this rotates it thirty degrees counter-clockwise around the vertical axis. If we apply $\mathbf{r}$ to the factor loading matrix of the model in the figure, we get the model in Figure 16.5. Now instead of X_1 being correlated with the other variables only through one factor, it's correlated through two factors, and X_4 has incoming arrows from three factors.

Because the transformation is orthogonal, the distribution of the observations is unchanged. In particular, the fit of the new factor model to the data will be *exactly* as good as the fit of the old model. If we try to take this causally, however, we come up with a very different interpretation. The quality of the fit to the data does not, therefore, let us distinguish between these two models, and so these two stories about the causal structure of the data.[14]

The rotation problem does not rule out the idea that checking the fit of a factor model would let us discover *how many* hidden causal variables there are.

[14] There might, of course, be other considerations, beyond the quality of the fit to the data, which would favor one model over another. This would have to be something like independent scientific evidence in favor of thinking that (say) X_1 only reflected a single latent variable. Note however that those other considerations could hardly be previous factor analyses of the same (or similar) variables, since they'd all be subject to the rotation problem as well.

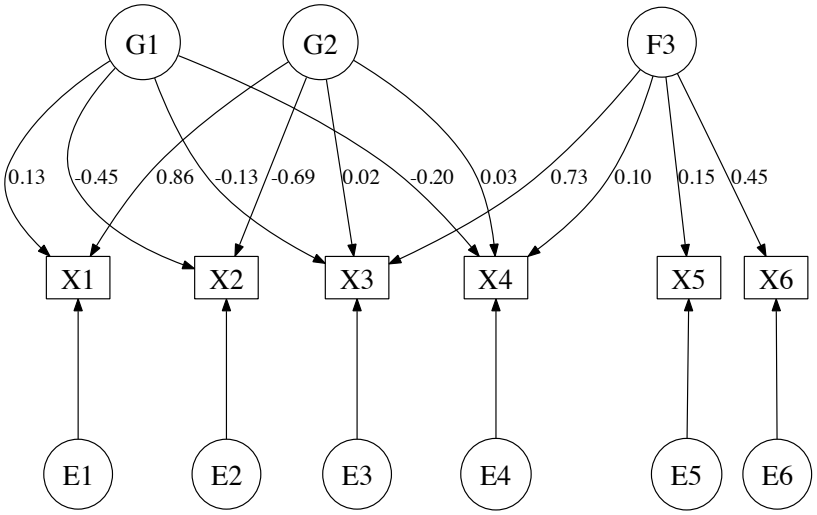


Figure 16.5 The model from Figure 16.1, after rotating the first two factors by 30 degrees around the third factor's axis. The new factor loadings are rounded to two decimal places.

16.9.2 Factors or Mixtures?

Suppose we have two distributions with probability densities $f_0(x)$ and $f_1(x)$. Then we can define a new distribution which is a **mixture** of them, with density $f_\alpha(x) = (1-\alpha)f_0(x) + \alpha f_1(x)$, $0 \leq \alpha \leq 1$. The same idea works if we combine more than two distributions, so long as the sum of the **mixing weights** sum to one (as do α and $1-\alpha$). Mixture models are a very flexible and useful way of representing complicated probability distributions[15], and we will look at them in detail in Chapter 17.

I bring up mixture models here because there is a very remarkable result: any linear factor model with q factors is equivalent to some mixture model with $q+1$ clusters, in the sense that the two models have the same means and covariances (Bartholomew, 1987, pp. 36–38). Recall from above that the likelihood of a factor model depends on the data only through the correlation matrix. If the data really

[15] They are also a probabilistic, predictive alternative to the kind of clustering techniques you may have seen in data mining, such as k-means: each distribution in the mixture is basically a cluster, and the mixing weights are the probabilities of drawing a new sample from the different clusters.

were generated by drawing from $q+1$ clusters, then a model with q factors can match the covariance matrix very well, and so get a very high likelihood. This means it will, by the usual test, seem like a very good fit. Needless to say, however, the causal interpretations of the mixture model and the factor model are very different. The two *may* be distinguishable if the clusters are well-separated (by looking to see whether the data are unimodal or not), but that's not exactly guaranteed.

All of which suggests that factor analysis can't alone really tell us whether we have q continuous latent variables, or one discrete hidden variable taking $q+1$ values.

16.9.3 The Thomson Sampling Model

We have been working with fewer factors than we have features. Suppose that's not true. Suppose that each of our features is actually a linear combination of a *lot* of variables we don't measure:

$$X_{ij} = \eta_{ij} + \sum_{k=1}^{q} A_{ik} T_{kj} = \eta_{ij} + \vec{A}_i \cdot \vec{T}_j \tag{16.49}$$

where $q \gg p$. Suppose further that the latent variables A_{ik} are totally independent of one another, but they all have mean 0 and variance 1; and that the noises η_{ij} are independent of each other and of the A_{ik}, with variance ϕ_j; and the T_{kj} are independent of everything. What then is the covariance between X_{ia} and X_{ib}? Well, because $\mathbb{E}[X_{ia}] = \mathbb{E}[X_{ib}] = 0$, it will just be the expectation of the product of the features:

$$\mathbb{E}[X_{ia} X_{ib}] \tag{16.50}$$
$$= \mathbb{E}\left[(\eta_{ia} + \vec{A}_i \cdot \vec{T}_a)(\eta_{ib} + \vec{A}_i \cdot \vec{T}_b)\right] \tag{16.51}$$
$$= \mathbb{E}[\eta_{ia}\eta_{ib}] + \mathbb{E}\left[\eta_{ia}\vec{A}_i \cdot \vec{T}_b\right] + \mathbb{E}\left[\eta_{ib}\vec{A}_i \cdot \vec{T}_a\right] + \mathbb{E}\left[(\vec{A}_i \cdot \vec{T}_a)(\vec{A}_i \cdot \vec{T}_b)\right] \tag{16.52}$$
$$= 0 + 0 + 0 + \mathbb{E}\left[\left(\sum_{k=1}^{q} A_{ik} T_{ka}\right)\left(\sum_{l=1}^{q} A_{il} T_{lb}\right)\right] \tag{16.53}$$
$$= \mathbb{E}\left[\sum_{k,l} A_{ik} A_{il} T_{ka} T_{lb}\right] \tag{16.54}$$
$$= \sum_{k,l} \mathbb{E}[A_{ik} A_{il}]\, T_{ka} T_{lb} \tag{16.55}$$
$$= \sum_{k,l} \mathbb{E}[A_{ik} A_{il}]\, \mathbb{E}[T_{ka} T_{lb}] \tag{16.56}$$
$$= \sum_{k=1}^{q} \mathbb{E}[T_{ka} T_{kb}] \tag{16.57}$$

where to get the last line I use the fact that $\mathbb{E}[A_{ik}A_{il}] = 1$ if $k = l$ and $= 0$ otherwise. If the coefficients T are fixed, then the last expectation goes away and we merely have the same kind of sum we've seen before, in the factor model.

Instead, however, let's say that the coefficients T are themselves random (but independent of A and η). For each feature X_{ia}, we fix a proportion z_a between 0 and 1. We then set $T_{ka} \sim \text{Bernoulli}(z_a)$, with $T_{ka} \perp\!\!\!\perp T_{lb}$ unless $k = l$ and $a = b$. Then

$$\mathbb{E}[T_{ka}T_{kb}] = \mathbb{E}[T_{ka}]\,\mathbb{E}[T_{kb}] = z_a z_b \tag{16.58}$$

and

$$\mathbb{E}[X_{ia}X_{ib}] = q z_a z_b \tag{16.59}$$

Of course, in the one-factor model,

$$\mathbb{E}[X_{ia}X_{ib}] = w_a w_b \tag{16.60}$$

So this random-sampling model looks *exactly* like the one-factor model with factor loadings proportional to z_a. The tetrad equation, in particular, will hold.

Now, it doesn't make a lot of sense to imagine that every time we make an observation we change the coefficients T randomly. Instead, let's suppose that they are first generated randomly, giving values T_{kj}, and then we generate feature values according to Eq. 16.49. The covariance between X_{ia} and X_{ib} will be $\sum_{k=1}^{q} T_{ka}T_{kb}$. But this is a sum of IID random values, so by the law of large numbers as q gets large this will become very close to $q z_a z_b$. Thus, for nearly all choices of the coefficients, the feature covariance matrix should come very close to satisfying the tetrad equations and looking like there's a single general factor.

In this model, each feature is a linear combination of a *random sample* of a huge pool of completely independent features, plus some extra noise specific to the feature.[16] Precisely *because* of this, the features are correlated, and the pattern of correlations is that of a factor model with one factor. The appearance of a single common cause actually arises from the fact that the number of causes is immense, and there is no particular pattern to their influence on the features.

Code Example 30 simulates the Thomson model.

```
tm <- rthomson(50,11,500,50)
factanal(tm$data,1)
##
## Call:
```

[16] When Godfrey Thomson introduced this model in 1914, he used a slightly different procedure to generate the coefficient T_{kj}. For each feature he drew a uniform integer between 1 and q, call it q_j, and then sampled the integers from 1 to q *without replacement* until he had q_j random numbers; these were the values of k where $T_{kj} = 1$. This is basically similar to what I describe, setting $z_j = q_j/q$, but a bit harder to analyze in an elementary way. — Thomson (1916), the original paper, includes what we would now call a simulation study of the model, where Thomson stepped through the procedure to produce simulated data, calculate the empirical correlation matrix of the features, and check the fit to the tetrad equations. Not having a computer, Thomson generated the values of T_{kj} with a deck of cards, and of the A_{ik} and η_{ij} by rolling *5220* dice.

```
# Simulate Godfrey Thomson's 'sampling model' of mental abilities, and perform
# factor analysis on the resulting test scores.

# Simulate the Thomson model Follow Thomson's original
# sampling-without-replacement scheme Pick a random number in 1:a for the
# number of shared abilities for each test Then draw a
# sample-without-replacement of that size from 1:a; those are the shared
# abilities summed in that test.  Specific variance of each test is also
# random; draw a number in 1:q, and sum that many independent normals, with the
# same parameters as the abilities.  Inputs: number of testees (n) number of
# tests (d) number of shared abilities (a) number of specific abilities per
# test (q) mean of each ability (mean) sd of each ability (sd) Depends on:
# mvrnorm from library MASS (multivariate random normal generator) Output:
# list, containing: matrix of test loadings on to general abilities vector of
# number of specific abilities per test matrix of abilities-by-testees matrix
# of general+specific scores by testees raw data (including measurement noise)
rthomson <- function(n, d, a, q, ability.mean = 0, ability.sd = 1) {
    # ATTN: Should really use more intuitive argument names number of testees =
    # n number of tests = d number of shared abilities = a max. number of
    # specific abilities per test = q

    stopifnot(require(MASS))  # for multivariate normal generation

    # assign abilities to tests
    general.per.test <- sample(1:a, size = d, replace = TRUE)
    specifics.per.test <- sample(1:q, size = d, replace = TRUE)

    # Define the matrix assigning abilities to tests
    general.to.tests <- matrix(0, a, d)
    # Exercise to the reader: Vectorize this
    for (i in 1:d) {
        abilities <- sample(1:a, size = general.per.test[i], replace = FALSE)
        general.to.tests[abilities, i] <- 1
    }

    # Covariance matrix of the general abilities
    sigma <- matrix(0, a, a)
    diag(sigma) <- (ability.sd)^2
    mu <- rep(ability.mean, a)
    x <- mvrnorm(n, mu, sigma)  # person-by-abilities matrix of abilities

    # The 'general' part of the tests
    general.tests <- x %*% general.to.tests
    # Now the 'specifics'
    specific.tests <- matrix(0, n, d)
    noisy.tests <- matrix(0, n, d)
    # Each test gets its own specific abilities, which are independent for each
    # person Exercise to the reader: vectorize this, too
    for (i in 1:d) {
        # Each test has noises.per.test disturbances, each of which has the
        # given sd; since these are all independent their variances add
        j <- specifics.per.test[i]
        specifics <- rnorm(n, mean = ability.mean * j, sd = ability.sd * sqrt(j))
        specific.tests[, i] <- general.tests[, i] + specifics
        # Finally, for extra realism, some mean-zero trial-to-trial noise, so
        # that if we re-use this combination of general and specific ability
        # scores, we won't get the exact same test scores twice
        noises <- rnorm(n, mean = 0, sd = ability.sd)
        noisy.tests[, i] <- specific.tests[, i] + noises
    }

    tm <- list(data = noisy.tests, general.ability.pattern = general.to.tests, numbers.of.specifics
        ability.matrix = x, specific.tests = specific.tests)
    return(tm)
}
```

Code Example 30: *Function for simulating the Thomson latent-sampling model.*

```
## factanal(x = tm$data, factors = 1)
##
## Uniquenesses:
##  [1] 0.142 0.102 0.083 0.798 0.862 0.665 0.076 0.872 0.523 0.356 0.714
##
## Loadings:
##       Factor1
##  [1,] 0.926
##  [2,] 0.947
##  [3,] 0.957
##  [4,] 0.449
##  [5,] 0.371
##  [6,] 0.579
##  [7,] 0.961
##  [8,] 0.358
##  [9,] 0.691
## [10,] 0.802
## [11,] 0.535
##
##                Factor1
## SS loadings      5.805
## Proportion Var   0.528
##
## Test of the hypothesis that 1 factor is sufficient.
## The chi square statistic is 59.24 on 44 degrees of freedom.
## The p-value is 0.0622
```

The first command generates data from $n = 50$ items with $p = 11$ features and $q = 500$ latent variables. (The last argument controls the average size of the specific variances ϕ_j.) The result of the factor analysis is of course variable, depending on the random draws; this attempt gave the proportion of variance associated with the factor as 0.53, and the p-value as 0.062. Repeating the simulation many times, one sees that the p-value is pretty close to uniformly distributed, which is what it should be if the null hypothesis is true (Figure 16.6). For fixed n, the distribution becomes closer to uniform the larger we make q. In other words, the goodness-of-fit test has little or no power against the alternative of the Thomson model.

Modifying the Thomson model to look like multiple factors grows notationally cumbersome; the basic idea however is to use multiple pools of independently-sampled latent variables, and sum them:

$$X_{ij} = \eta_{ij} + \sum_{k=1}^{q_1} A_{ik} T_{kj} + \sum_{k=1}^{q_2} B_{ik} R_{kj} + \ldots \tag{16.61}$$

where the T_{kj} coefficients are uncorrelated with the R_{kj}, and so forth. In expectation, if there are r such pools, this *exactly* matches the factor model with r factors, and any particular realization is overwhelmingly likely to match if the $q_1, q_2, \ldots q_r$ are large enough.[17]

[17] A recent paper on the Thomson model (Bartholomew *et al.*, 2009) proposes just this modification to multiple factors and to Bernoulli sampling. However, I proposed this independently, in the fall 2008 version of these notes, about a year before their paper.

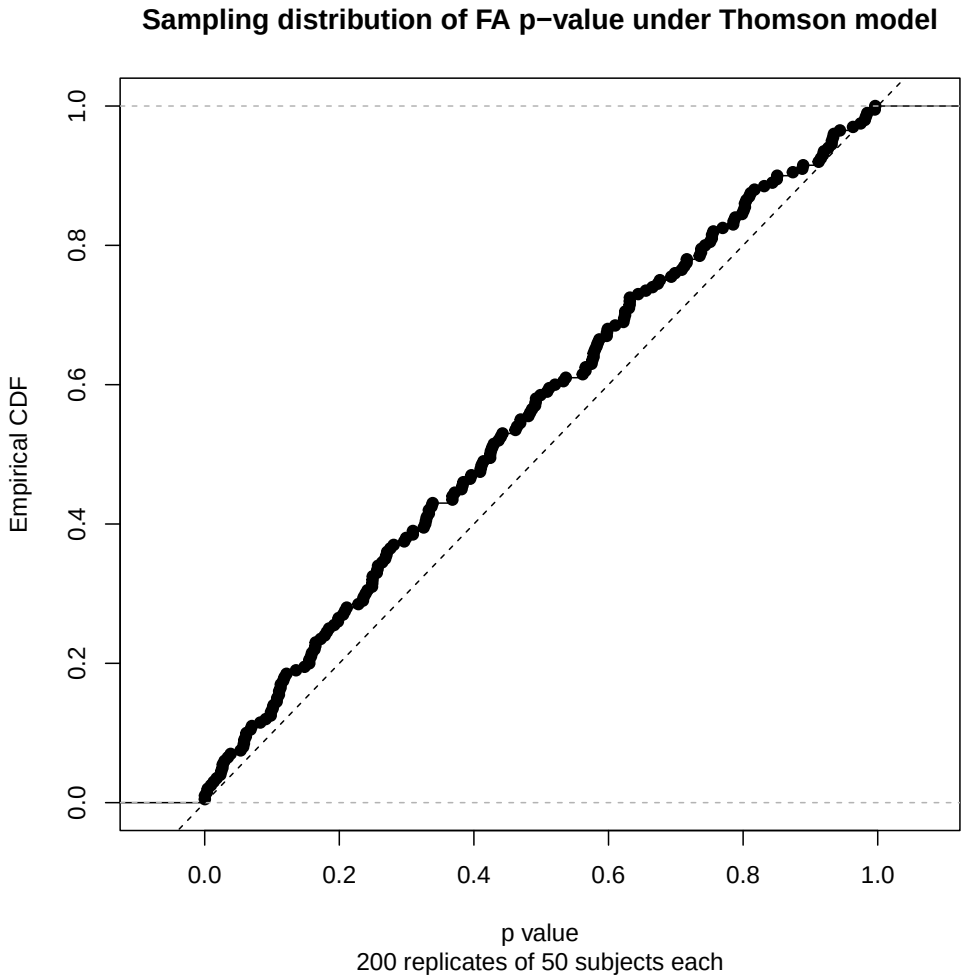


Figure 16.6 Mimcry of the one-factor model by the Thomson model. The Thomson model was simulated 200 times with the parameters given above; each time, the simulated data was then fit to a factor model with one factor, and the p-value of the goodness-of-fit test extracted. The plot shows the empirical cumulative distribution function of the p-values. If the null hypothesis were exactly true, then $p \sim \text{Unif}(0,1)$, and the theoretical CDF would be the diagonal line (dashed).

It's not feasible to estimate the T of the Thomson model in the same way that we estimate factor loadings, because $q > p$. This is not the point of considering the model, which is rather to make it clear that we actually learn very little about where the data come from when we learn that a factor model fits well. It could mean that the features arise from combining a small number of factors, or on the

contrary from combining a huge number of factors in a random fashion. A lot of the time the latter is a more plausible-sounding story.

For example, a common application of factor analysis is in marketing: you survey consumers and ask them to rate a bunch of products on a range of features, and then do factor analysis to find attributes which summarize the features. That's fine, but it may well be that each of the features is influenced by lots of aspects of the product you don't include in your survey, and the correlations are really explained by different features being affected by many of the same small aspects of the product. Similarly for psychological testing: answering any question is really a pretty complicated process involving lots of small processes and skills (of perception, several kinds of memory, problem-solving, attention, motivation, etc.), which overlap partially from question to question.

16.10 Further Reading

The classical papers by Spearman (1904) and Thurstone (1934) are readily available online, and very much worth reading for getting a sense of the problems which motivated the introduction of factor analysis, and the skill with which the founders grappled with them. Loehlin (1992) is a decent textbook intended for psychologists; the presumed mathematical and statistical level is decidedly lower than that of this book, but it's still useful. Thomson (1939) remains one of the most *insightful* books on factor analysis, though obviously there have been a lot of technical refinements since he wrote. It's strongly recommended for anyone who plans to make much use of the method. While out of print, used copies are reasonably plentiful and cheap, and at least one edition is free online.

On purely statistical issues related to factor analysis, Bartholomew (1987) is by far the best reference I have found; it quite properly sets it in the broader context of latent variable models, including the sort of latent class models we will explore in Chapter 17. The computational advice of that edition is, necessarily, now quite obsolete; there is an updated edition from 2011, which I have not been able to consult by the time of writing.

The use of factor analysis in psychological testing has given rise to a large controversial literature, full of claims, counter-claims, counter-counter-claims, and so on *ad nauseam*. Without, here, going into that, I will just note that to the extent the best arguments against (say) reifying the general factor extracted from IQ tests as "general intelligence" are good arguments, they do not just apply to intelligence, but also to personality tests, and indeed to many procedures outside psychology. In other words, if there's a problem, it's not *just* a problem for intelligence testing alone, or even for psychology alone. On this, see Glymour (1998) and Borsboom (2005, 2006).

Exercises

16.1 Prove Eq. 16.13.

16.2 Why is it fallacious to go from "the data have the kind of correlations predicted by a one-factor model" to "the data were generated by a one-factor model"?
16.3 Consider Figure 16.2 and its code. What is $\mathbf{w}$ here? What is ψ? What is the implied covariance matrix of $\vec{X}$? .
16.4 Show that the correlation between the jth feature and G, in the one-factor model, is w_j.
16.5 Check that Eq. 16.11 and Eq. 16.25 are compatible.
16.6 Find the weights b_{jr} for the Thomson estimator of factor scores (Eq. 16.38), assuming you know $\mathbf{w}$. Do you need to assume a Gaussian distribution?

17

Mixture Models

17.1 Two Routes to Mixture Models

17.1.1 From Factor Analysis to Mixture Models

In factor analysis, the origin myth is that we have a fairly small number, q of real variables which happen to be unobserved ("latent"), and the much larger number p of variables we do observe arise as linear combinations of these factors, plus noise. The mythology is that it's possible for us (or for Someone) to *continuously* adjust the latent variables, and the distribution of observables changes linearly in response. What if the latent variables are not continuous but ordinal, or even categorical? The natural idea would be that each value of the latent variable would give a different distribution of the observables.

17.1.2 From Kernel Density Estimates to Mixture Models

We have also previously looked at kernel density estimation, where we approximate the true distribution by sticking a small ($\frac{1}{n}$ weight) copy of a kernel pdf at each observed data point and adding them up. With enough data, this comes arbitrarily close to any (reasonable) probability density, but it does have some drawbacks. Statistically, it labors under the curse of dimensionality. Computationally, we have to remember *all* of the data points, which is a lot. We saw similar problems when we looked at fully non-parametric regression, and then saw that both could be ameliorated by using things like additive models, which impose more constraints than, say, unrestricted kernel smoothing. Can we do something like that with density estimation?

Additive modeling for densities is not as common as it is for regression — it's harder to think of times when it would be natural and well-defined[1] — but we can do things to restrict density estimation. For instance, instead of putting a copy of the kernel at *every* point, we might pick a small number $K \ll n$ of points, which we feel are somehow typical or representative of the data, and put a copy of the kernel at each one (with weight $\frac{1}{K}$). This uses less memory, but it

[1] Remember that the integral of a probability density over all space must be 1, while the integral of a regression function doesn't have to be anything in particular. If we had an additive density, $f(x) = \sum_j f_j(x_j)$, ensuring normalization is going to be very tricky; we'd need $\sum_j \int f_j(x_j) dx_1 dx_2 \ldots dx_p = 1$. It would be easier to ensure normalization while making the *log*-density additive, but that assumes the features are independent of each other.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

ignores the other data points, and lots of them are probably very similar to those points we're taking as prototypes. The differences between prototypes and many of their neighbors are just matters of chance or noise. Rather than remembering all of those noisy details, why not collapse those data points, and just remember their common distribution? Different regions of the data space will have different shared distributions, but we can just combine them.

17.1.3 Mixture Models

More formally, we say that a distribution f is a **mixture** of K **cluster**[2] distributions $f_1, f_2, \ldots f_K$ if

$$f(x) = \sum_{k=1}^{K} \lambda_k f_k(x) \tag{17.1}$$

with the λ_k being the **mixing weights**, $\lambda_k > 0$, $\sum_k \lambda_k = 1$. Eq. 17.1 is a complete stochastic model, so it gives us a recipe for generating new data points: first pick a distribution, with probabilities given by the mixing weights, and then generate one observation according to that distribution. Symbolically,

$$Z \sim \text{Mult}(\lambda_1, \lambda_2, \ldots \lambda_K) \tag{17.2}$$

$$X|Z \sim f_Z \tag{17.3}$$

where I've introduced the discrete random variable Z which says which cluster X is drawn from.

I haven't said what kind of distribution the f_ks are. In principle, we could make these completely arbitrary, and we'd still have a perfectly good mixture model. In practice, a lot of effort is given over to **parametric mixture** models, where the f_k are all from the same parametric family, but with different parameters — for instance they might all be Gaussians with different centers and variances, or all Poisson distributions with different means, or all power laws with different exponents. (It's not necessary, just customary, that they all be of the same kind.) We'll write the parameter, or parameter vector, of the kth cluster as θ_k, so the model becomes

$$f(x) = \sum_{k=1}^{K} \lambda_k f(x; \theta_k) \tag{17.4}$$

The over-all parameter vector of the mixture model is thus $\theta = (\lambda_1, \lambda_2, \ldots \lambda_K, \theta_1, \theta_2, \ldots \theta_K)$.

Let's consider two extremes. When $K = 1$, we have a simple parametric distribution, of the usual sort, and density estimation reduces to estimating the parameters, by maximum likelihood or whatever else we feel like. On the other hand when $K = n$, the number of observations, we have gone back towards kernel density estimation. If K is fixed as n grows, we still have a parametric model,

[2] Many people write "components" instead of "clusters", but I am deliberately avoiding that here so as not to lead to confusion with the components of PCA.

and avoid the curse of dimensionality, but a mixture of (say) ten Gaussians is more flexible than a single Gaussian — thought it may still be the case that the true distribution just can't be written as a ten-Gaussian mixture. So we have our usual bias-variance or accuracy-precision trade-off — using many clusters in the mixture lets us fit many distributions very accurately, with low approximation error or bias, but means we have more parameters and so we can't fit any one of them as precisely, and there's more variance in our estimates.

17.1.4 Geometry

In Chapter 15, we looked at principal components analysis, which finds linear structures with q dimensions (lines, planes, hyper-planes, ...) which are good approximations to our p-dimensional data, $q \ll p$. In Chapter 16, we looked at factor analysis, which imposes a statistical model for the distribution of the data around this q-dimensional plane (Gaussian noise), and a statistical model of the distribution of representative points on the plane (also Gaussian). This set-up is implied by the mythology of linear continuous latent variables, but can arise in other ways.

We know from geometry that it takes $q + 1$ points to define a q-dimensional plane, and that in general any $q+1$ points on the plane will do. This means that if we use a mixture model with $q+1$ clusters, we will also get data which lies around a q-dimensional plane. Furthermore, by adjusting the mean of each cluster, and their relative weights, we can make the global mean of the mixture whatever we like. And we can even match the covariance matrix of any q-factor model by using a mixture with $q + 1$ clusters[3]. Now, this mixture distribution will hardly ever be exactly the same as the factor model's distribution — mixtures of Gaussians aren't Gaussian, the mixture will usually (but not always) be multimodal while the factor distribution is always unimodal — but it will have the same geometry (q-dimensional subspace plus noise), and the same mean and the same covariances, so we will have to look beyond those to tell them apart. Which, frankly, people hardly ever do.

17.1.5 Identifiability

Before we set about trying to estimate our probability models, we need to make sure that they are identifiable — that if we have distinct parameter values in the model, we get distinct distributions over the observables. Sometimes we use too many parameters, or badly chosen parameters, and lose identifiability. If there *are* distinct representations which are observationally equivalent, we either need to change our model, change our representation, or fix on a *unique* representation by some convention. For example:

- With additive regression, $\mathbb{E}[Y|X = x] = \alpha + \sum_j f_j(x_j)$, we can add arbitrary constants so long as they cancel out. That is, we get the same predictions

[3] See Bartholomew (1987, pp. 36–38). The proof is tedious algebraically.

from $\alpha + c_0 + \sum_j f_j(x_j) + c_j$ when $c_0 = -\sum_j c_j$. This is another model of the same form, $\alpha' + \sum_j f'_j(x_j)$, so it's not identifiable. We dealt with this by imposing the convention that $\alpha = \mathbb{E}[Y]$ and $\mathbb{E}[f_j(X_j)] = 0$ — we picked out a favorite, convenient representation from the infinite collection of equivalent representations.

- Linear regression becomes unidentifiable with collinear features. Collinearity is a good reason to not use linear regression (i.e., we change the model.)
- Factor analysis is unidentifiable because of the rotation problem. Some people respond by trying to fix on a particular representation, others just ignore it.

Two kinds of identification problems are common for mixture models; one is trivial and the other is fundamental. The trivial one is that we can always swap the labels of any two clusters with no effect on anything observable at all — if we decide that cluster number 1 is now cluster number 7 and vice versa, that doesn't change the distribution of X at all. This **label switching** or **label degeneracy** can be annoying, especially for some estimation algorithms, but that's the worst of it.

A more fundamental lack of identifiability happens when mixing two distributions from a parametric family just gives us a third distribution from the same family. For example, suppose we have a single binary feature, say an indicator for whether someone will pay back a credit card. We might think there are two kinds of customers, with high- and low- risk of not paying, and try to represent this as a mixture of Bernoulli distribution. If we try this, we'll see that we've gotten a *single* Bernoulli distribution with an intermediate risk of repayment. A mixture of Bernoulli is always just another Bernoulli. More generally, a mixture of discrete distributions over any finite number of categories is just another distribution over those categories[4].

17.1.6 Probabilistic Clustering

Here is yet another way to view mixture models, which I hinted at when I talked about how they are a way of putting similar data points together into "clusters", where clusters are represented by the distributions going into the mixture. The idea is that all data points of the same type, belonging to the same cluster or class, are more or less equivalent and all come from the same distribution, and any differences between them are matters of chance. This view *exactly* corresponds to mixture models like Eq. 17.1; the hidden variable Z I introduced above in just the cluster label.

One of the very nice things about probabilistic clustering is that Eq. 17.1 actually *claims* something about what the data looks like; it says that it follows a

[4] That is, a mixture of any two $n = 1$ multinomials is another $n = 1$ multinomial. This is not generally true when $n > 1$; for instance, a mixture of a Binom$(2, 0.75)$ and a Binom$(2, 0.25)$ is *not* a Binom$(2, p)$ for any p (Exercise 17.2). However, both of those binomials is a distribution on $\{0, 1, 2\}$, and so is their mixture. This apparently trivial point actually leads into very deep topics, since it turns out that which models can be written as mixtures of others is strongly related to what properties of the data-generating process can actually be learned from data: see Lauritzen (1984).

certain distribution. We can check whether it does, and we can check whether *new* data follows this distribution. If it does, great; if not, if the predictions systematically fail, then the model is wrong. We can compare different probabilistic clusterings by how well they predict (say under cross-validation).[5]

In particular, probabilistic clustering gives us a sensible way of answering the question "how many clusters?" The best number of clusters to use is the number which will best generalize to future data. If we don't want to wait around to get new data, we can approximate generalization performance by cross-validation, or by any other adaptive model selection procedure.

17.1.7 Simulation

Simulating from a mixture model works rather like simulating from a kernel density estimate (§14.7.1). To draw a new value $\tilde{X}$, first draw a random integer Z from 1 to k, with probabilities λ_k, then draw from the Zth cluster. (That is, $\tilde{X}|Z \sim f_Z$.) Note that if we want multiple draws, $\tilde{X}_1, \tilde{X}_2, \ldots \tilde{X}_b$, each of them needs an independent Z.

17.2 Estimating Parametric Mixture Models

From intro stats., we remember that it's generally a good idea to estimate distributions using maximum likelihood, when we can. How could we do that here?

Remember that the likelihood is the probability (or probability density) of observing our data, as a function of the parameters. Assuming independent samples, that would be

$$\prod_{i=1}^{n} f(x_i; \theta) \tag{17.5}$$

for observations $x_1, x_2, \ldots x_n$. As always, we'll use the logarithm to turn multiplication into addition:

$$\ell(\theta) = \sum_{i=1}^{n} \log f(x_i; \theta) \tag{17.6}$$

$$= \sum_{i=1}^{n} \log \sum_{k=1}^{K} \lambda_k f(x_i; \theta_k) \tag{17.7}$$

[5] Contrast this with k-means or hierarchical clustering, which you may have seen in other classes: they make no predictions, and so we have no way of telling if they are right or wrong. Consequently, comparing different non-probabilistic clusterings is a lot harder!

Let's try taking the derivative of this with respect to one parameter, say θ_j.

$$\frac{\partial \ell}{\partial \theta_j} = \sum_{i=1}^{n} \frac{1}{\sum_{k=1}^{K} \lambda_k f(x_i; \theta_k)} \lambda_j \frac{\partial f(x_i; \theta_j)}{\partial \theta_j} \tag{17.8}$$

$$= \sum_{i=1}^{n} \frac{\lambda_j f(x_i; \theta_j)}{\sum_{k=1}^{K} \lambda_k f(x_i; \theta_k)} \frac{1}{f(x_i; \theta_j)} \frac{\partial f(x_i; \theta_j)}{\partial \theta_j} \tag{17.9}$$

$$= \sum_{i=1}^{n} \frac{\lambda_j f(x_i; \theta_j)}{\sum_{k=1}^{K} \lambda_k f(x_i; \theta_k)} \frac{\partial \log f(x_i; \theta_j)}{\partial \theta_j} \tag{17.10}$$

If we just had an ordinary parametric model, on the other hand, the derivative of the log-likelihood would be

$$\sum_{i=1}^{n} \frac{\partial \log f(x_i; \theta_j)}{\partial \theta_j} \tag{17.11}$$

So maximizing the likelihood for a mixture model is like doing a *weighted* likelihood maximization, where the weight of x_i depends on cluster, being

$$w_{ij} = \frac{\lambda_j f(x_i; \theta_j)}{\sum_{k=1}^{K} \lambda_k f(x_i; \theta_k)} \tag{17.12}$$

The problem is that these weights depend on the parameters we are trying to estimate![6]

Let's look at these weights w_{ij} a bit more. Remember that λ_j is the probability that the hidden class variable Z is j, so the numerator in the weights is the joint probability of getting $Z = j$ and $X = x_i$. The denominator is the marginal probability of getting $X = x_i$, so the ratio is the conditional probability of $Z = j$ given $X = x_i$,

$$w_{ij} = \frac{\lambda_j f(x_i; \theta_j)}{\sum_{k=1}^{K} \lambda_k f(x_i; \theta_k)} = p(Z = j | X = x_i; \theta) \tag{17.13}$$

If we try to estimate the mixture model, then, we're doing weighted maximum likelihood, with weights given by the posterior cluster probabilities. These, to repeat, depend on the parameters we are trying to estimate, so there seems to be a vicious circle.

But, as the saying goes, one man's vicious circle is another man's successive approximation procedure. A crude way of doing this[7] would start with an initial guess about the cluster distributions; find out which cluster each point is most likely to have come from; re-estimate the clusters using only the points assigned to it, etc., until things converge. This corresponds to taking all the weights w_{ij} to be either 0 or 1. However, it does not maximize the likelihood, since we've seen that to do so we need fractional weights.

What's called the EM algorithm is simply the obvious refinement of this "hard" assignment strategy.

[6] Matters are no better, but also no worse, for finding λ_j; see Exercise 17.3.

[7] Related to what's called "k-means" clustering.

1. Start with guesses about the cluster distributions $\theta_1, \theta_2, \ldots \theta_K$ and the mixing weights $\lambda_1, \ldots \lambda_K$.
2. Until nothing changes very much:
   1. Using the current parameter guesses, calculate the weights w_{ij} (E-step)
   2. Using the current weights, maximize the weighted likelihood to get new parameter estimates (M-step)
3. Return the final parameter estimates (including mixing proportions) and cluster probabilities

The M in "M-step" and "EM" stands for "maximization", which is pretty transparent. The E stands for "expectation", because it gives us the conditional probabilities of different values of Z, and probabilities are expectations of indicator functions. (In fact in some early applications, Z was binary, so one really was computing the expectation of Z.) The whole thing is thus formally the "expectation-maximization" algorithm, but "EM" is more common.

17.2.1 More about the EM Algorithm

The EM algorithm turns out to be a general way of maximizing the likelihood when some variables are unobserved, and hence useful for other things besides mixture models (e.g., when some variables are missing some of the time — see App. I.3.2). So in this section, where I try to explain why it works, I am going to be a bit more general and abstract. (Also, it will actually cut down on notation.) I'll pack the whole sequence of observations $x_1, x_2, \ldots x_n$ into a single variable d (for "data"), and likewise the whole sequence of $z_1, z_2, \ldots z_n$ into h (for "hidden"). What we want to do is maximize

$$\ell(\theta) = \log p(d;\theta) = \log \sum_h p(d,h;\theta) \tag{17.14}$$

This is generally hard, because even if $p(d,h;\theta)$ has a nice parametric form, that is lost when we sum up over all possible values of h (as we saw above for mixture models). The essential trick of the EM algorithm is to maximize not the log likelihood, but a *lower bound* on the log-likelihood, which is more tractable; we'll see that this lower bound is sometimes tight, i.e., coincides with the actual log-likelihood, and in particular does so at the global optimum.

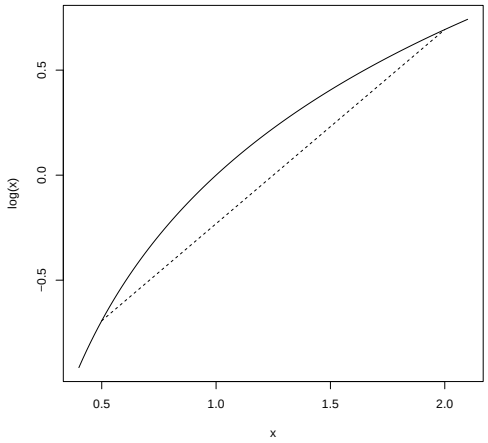
We can introduce an arbitrary[8] distribution on h, call it $q(h)$, and we'll write

$$\ell(\theta) = \log \sum_h p(d,h;\theta) \tag{17.15}$$

$$= \log \sum_h \frac{q(h)}{q(h)} p(d,h;\theta) \tag{17.16}$$

$$= \log \sum_h q(h) \frac{p(d,h;\theta)}{q(h)} \tag{17.17}$$

[8] Well, almost arbitrary; if some h has probability > 0 for all θ, then it shouldn't give that h probability zero.




```
curve(log(x),from=0.4,to=2.1)
segments(0.5,log(0.5),2,log(2),lty=2)
```

Figure 17.1 The logarithm is a concave function, i.e., the curve connecting any two points lies above the straight line doing so. Thus the average of logarithms is less than the logarithm of the average.

So far so trivial.

Now we need a geometric fact about the logarithm function, which is that its curve is concave: if we take any two points on the curve and connect them by a straight line, the curve lies above the line (Figure 17.1 and Exercise 17.6). Algebraically, this means that

$$w \log t_1 + (1-w) \log t_2 \leq \log wt_1 + (1-w)t_2 \tag{17.18}$$

for any $0 \leq w \leq 1$, and any points $t_1, t_2 > 0$. Nor does this just hold for two points: for any r points $t_1, t_2, \ldots t_r > 0$, and any set of non-negative weights $\sum_{i=1}^{r} w_r = 1$,

$$\sum_{i=1}^{r} w_i \log t_i \leq \log \sum_{i=1}^{r} w_i t_i \tag{17.19}$$

In words: the log of the average is at least the average of the logs. This is called **Jensen's inequality** . So

$$\log \sum_h q(h) \frac{p(d,h;\theta)}{q(h)} \geq \sum_h q(h) \log \frac{p(d,h;\theta)}{q(h)} \tag{17.20}$$

$$\equiv J(q,\theta) \tag{17.21}$$

We bother with all this because we hope that it will be easier to maximize this lower bound on the likelihood than the actual likelihood, and further hope

that the lower bound is reasonably tight. As to tightness, suppose that we set $q(h) = p(h|d;\theta)$. For this special choice of q, call it $\hat{q}$,

$$\frac{p(d,h;\theta)}{\hat{q}(h)} = \frac{p(d,h;\theta)}{p(h|d;\theta)} = \frac{p(d,h;\theta)}{p(h,d;\theta)/p(d;\theta)} = p(d;\theta) \tag{17.22}$$

no matter what h is. This implies $J(\hat{q},\theta) = \ell(\theta)$:

$$J(\hat{q},\theta) = \sum_h \hat{q}(h) \log \frac{p(d,h;\theta)}{\hat{q}(h)} \tag{17.23}$$

$$= \sum_h p(h|d;\theta) \log p(d;\theta) \tag{17.24}$$

$$= \log p(d;\theta) \sum_h p(h|d;\theta) \tag{17.25}$$

$$= \ell(\theta) \tag{17.26}$$

using Eq. 17.22 in the second line. This means that the lower bound $J(q,\theta) \leq \ell(\theta)$ is tight. Moreover, setting $q = \hat{q}$ maximizes $J(q,\theta)$ for fixed θ.

Here's how the EM algorithm goes in this formulation.

1. Start with an initial guess $\theta^{(0)}$ about the clusters and mixing weights.
2. Until nothing changes very much
   1. **E-step**: $q^{(t)} = \text{argmax}_q J(q,\theta^{(t)})$, i.e., set $q^{(t)}(h) = p(h|d;\theta^{(t)})$.
   2. **M-step**: $\theta^{(t+1)} = \text{argmax}_\theta J(q^{(t)},\theta)$
3. Return final estimates of θ and q

The E and M steps are now nice and symmetric; both are about maximizing J. It's easy to see that, after the E step,

$$J(q^{(t)},\theta^{(t)}) \geq J(q^{(t-1)},\theta^{(t)}) \tag{17.27}$$

and that, after the M step,

$$J(q^{(t)},\theta^{(t+1)}) \geq J(q^{(t)},\theta^{(t)}) \tag{17.28}$$

Putting these two inequalities together,

$$J(q^{(t+1)},\theta^{(t+1)}) \geq J(q^{(t)},\theta^{(t)}) \tag{17.29}$$

$$\ell(\theta^{(t+1)}) \geq \ell(\theta^{(t)}) \tag{17.30}$$

So each EM iteration can only improve the likelihood, guaranteeing convergence to a local maximum. Since it only guarantees a local maximum, it's a good idea to try a few different initial values of $\theta^{(0)}$ and take the best.

We saw above that the maximization in the E step is just computing the posterior probability $p(h|d;\theta)$. What about the maximization in the M step?

$$\sum_h q(h) \log \frac{p(d,h;\theta)}{q(h)} = \sum_h q(h) \log p(d,h;\theta) - \sum_h q(h) \log q(h) \tag{17.31}$$

The second sum doesn't depend on θ at all, so it's irrelevant for maximizing,

giving us back the optimization problem from the last section. This confirms that using the lower bound from Jensen's inequality hasn't yielded a different algorithm! (Exercise 17.10)

17.2.2 Topic Models and Probabilistic LSA

Mixture models over words provide an alternative to latent semantic indexing (§15.4) for document analysis. Instead of finding the principal components of the bag-of-words vectors, the idea is as follows. There are a certain number of **topics** which documents in the corpus can be about; each topic corresponds to a distribution over words. The distribution of words in a document is a mixture of the topic distributions. That is, one can generate a bag of words by first picking a topic according to a multinomial distribution (topic i occurs with probability λ_i), and then picking a word from that topic's distribution. The distribution of topics varies from document to document, and this is what's used, rather than projections on to the principal components, to summarize the document. This idea was, so far as I can tell, introduced by Hofmann (1999), who estimated everything by EM. **Latent Dirichlet allocation**, due to Blei and collaborators (Blei *et al.*, 2003) is an important variation which smoothes the topic distributions; there is a CRAN package called `lda`. Blei and Lafferty (2009) is a good review paper of the area.

17.3 Non-parametric Mixture Modeling

We could replace the M step of EM by some other way of estimating the distribution of each cluster. This could be a fast-but-crude estimate of parameters (say a method-of-moments estimator if that's simpler than the MLE), or it could even be a non-parametric density estimator of the type we talked about in Chapter 14. (Similarly for mixtures of regressions, etc.) Issues of dimensionality re-surface now, as well as convergence: because we're not, in general, increasing J at each step, it's harder to be sure that the algorithm will in fact converge. This is an active area of research.

17.4 Worked Computing Example: Snoqualmie Falls Revisited

17.4.1 Mixture Models in R

There are several R packages which implement mixture models. The `mclust` package (`http://www.stat.washington.edu/mclust/`) is pretty much standard for Gaussian mixtures. One of the more recent and powerful is `mixtools` (Benaglia *et al.*, 2009), which, in addition to classic mixtures of parametric densities, handles mixtures of regressions and some kinds of non-parametric mixtures. The `FlexMix` package (Leisch, 2004) is (as the name implies) very good at flexibly handling complicated situations, though you have to do some programming to take advantage of this.

17.4.2 Fitting a Mixture of Gaussians to Real Data

Let's go back to the Snoqualmie Falls data set, last used in §11.7[9]. There we built a system to forecast whether there would be precipitation on day t, on the basis of how much precipitation there was on day $t-1$. Let's look at the distribution of the amount of precipitation on the wet days.

```
snoqualmie <- scan("http://www.stat.washington.edu/peter/book.data/set1",skip=1)
snoq <- snoqualmie[snoqualmie > 0]
```

Figure 17.2 shows a histogram (with a fairly large number of bins), together with a simple kernel density estimate. This suggests that the distribution is rather skewed to the right, which is reinforced by the simple summary statistics:

```
summary(snoq)
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1.00    6.00   19.00   32.28   44.00  463.00
```

Notice that the mean is larger than the median, and that the distance from the first quartile to the median is much smaller (13/100 of an inch of precipitation) than that from the median to the third quartile (25/100 of an inch). One way this could arise, of course, is if there are multiple types of wet days, each with a different characteristic distribution of precipitation.

We'll look at this by trying to fit Gaussian mixture models with varying numbers of clusters. We'll start by using a mixture of two Gaussians. We *could* code up the EM algorithm for fitting this mixture model from scratch, but instead we'll use the `mixtools` package.

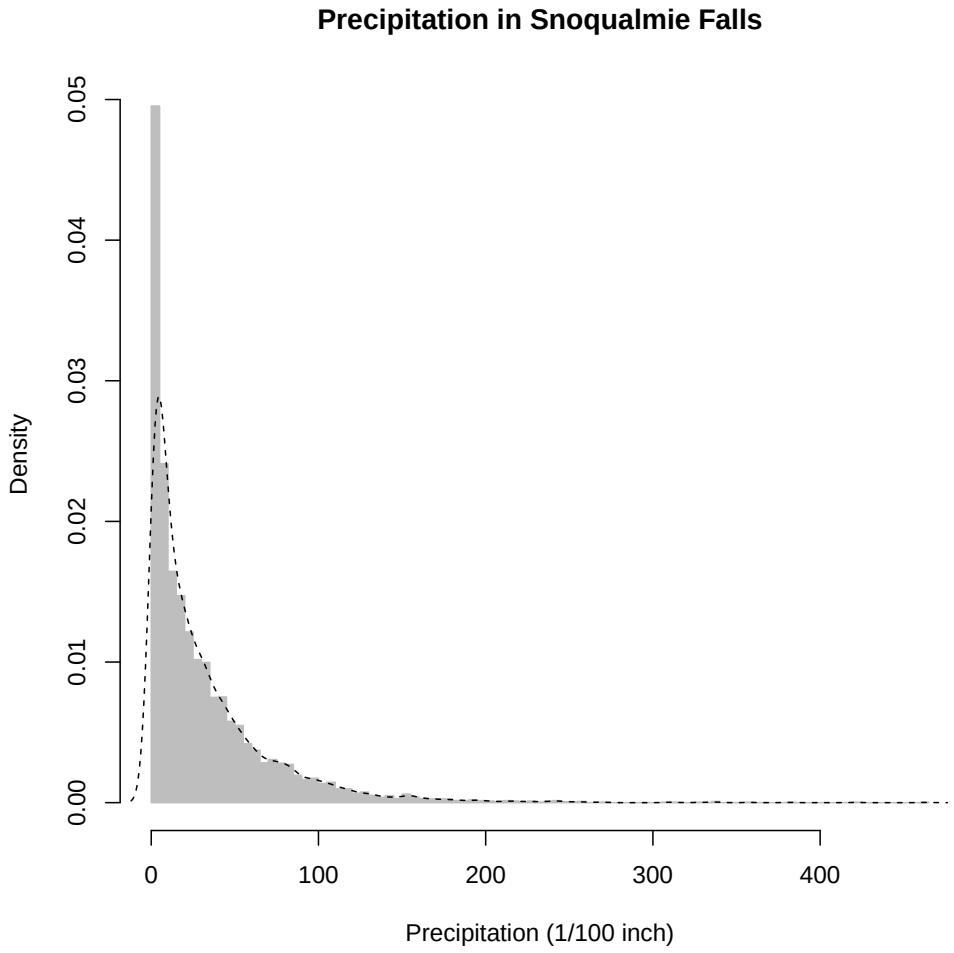
```
library(mixtools)
snoq.k2 <- normalmixEM(snoq,k=2,maxit=100,epsilon=0.01)
```

The EM algorithm "runs until convergence", i.e., until things change so little that we don't care any more. For the implementation in `mixtools`, this means running until the log-likelihood changes by less than `epsilon`. The default tolerance for convergence is not 10^{-2}, as here, but 10^{-8}, which can take a very long time indeed. The algorithm also stops if we go over a maximum number of iterations, even if it has not converged, which by default is 1000; here I have dialed it down to 100 for safety's sake. What happens?

```
snoq.k2 <- normalmixEM(snoq,k=2,maxit=100,epsilon=0.01)
```

```
summary(snoq.k2)
## summary of normalmixEM object:
##          comp 1   comp 2
## lambda  0.55734  0.44266
## mu     10.26065 59.99530
## sigma   8.50508 44.99334
## loglik at estimate:  -32681.21
```

[9] See that section for explanations of some of the data manipulation done in this section.




```
plot(hist(snoq,breaks=101),col="grey",border="grey",freq=FALSE,
     xlab="Precipitation (1/100 inch)",main="Precipitation in Snoqualmie Falls")
lines(density(snoq),lty="dashed")
```

Figure 17.2 Histogram (grey) for precipitation on wet days in Snoqualmie Falls. The dashed line is a kernel density estimate, which is not completely satisfactory. (It gives non-trivial probability to negative precipitation, for instance.)

There are two clusters, with weights (`lambda`) of about 0.56 and 0.44, two means (`mu`) and two standard deviations (`sigma`). The over-all log-likelihood, obtained after 59 iterations, is -3.2681214×10^4. (Demanding convergence to

$\pm 10^{-8}$ would thus have required the log-likelihood to change by less than one part in a trillion, which is quite excessive when we only have 6920 observations.)

We can plot this along with the histogram of the data and the non-parametric density estimate. I'll write a little function for it.

```
# Plot the (scaled) density associated with a Gaussian cluster
# Inputs: mixture object (mixture)
  # index number of the cluster (cluster.number)
  # optional additional arguments to curve (...)
# Outputs: None useful
# Side-effects: Plot is added to the current display
plot.gaussian.clusters <- function(mixture, cluster.number, ...) {
  curve(mixture$lambda[cluster.number] *
        dnorm(x,mean=mixture$mu[cluster.number],
        sd=mixture$sigma[cluster.number]), add=TRUE, ...)
}
```

This adds the density of a given cluster to the current plot, but scaled by the share it has in the mixture, so that it is visually comparable to the over-all density.

17.4.3 Calibration-checking for the Mixture

Examining the two-cluster mixture, it does not look altogether satisfactory — it seems to consistently give too much probability to days with about 1 inch of precipitation. Let's think about how we could check things like this.

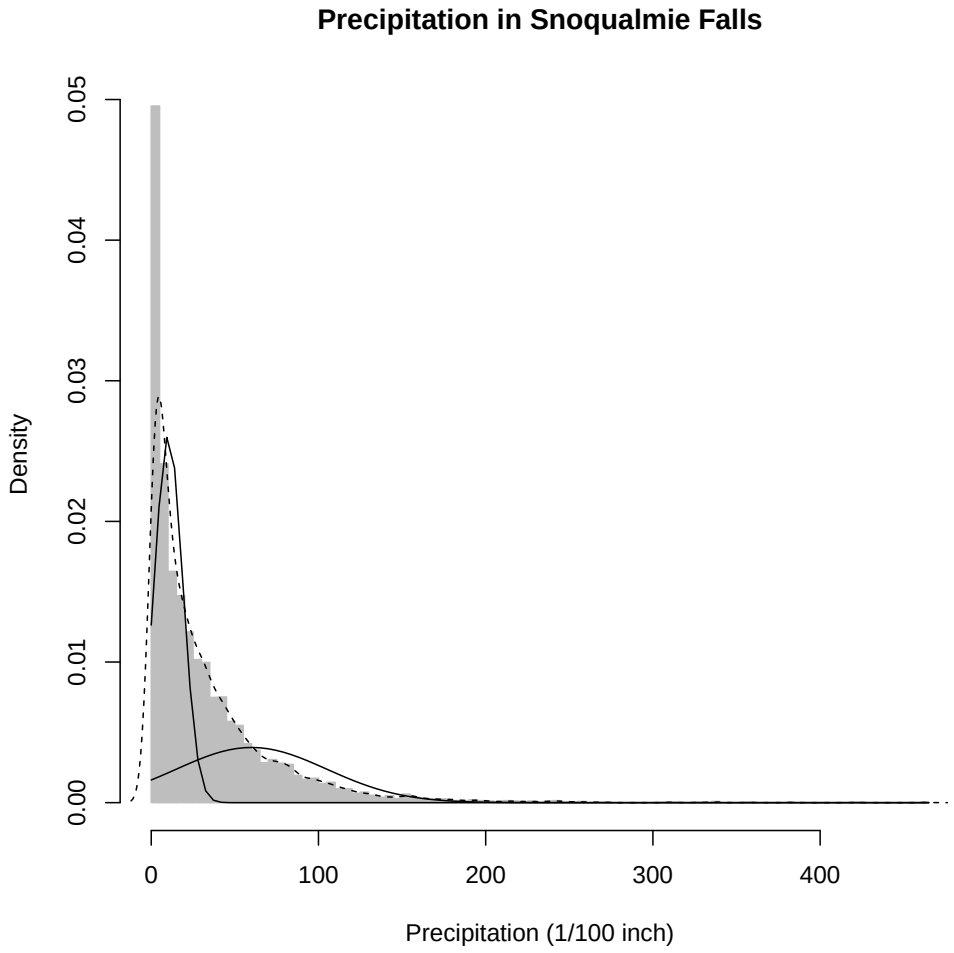
When we looked at logistic regression, we saw how to check probability forecasts by checking calibration — events predicted to happen with probability p should in fact happen with frequency $\approx p$. Here we don't have a binary event, but we do have lots of probabilities. In particular, we have a cumulative distribution function $F(x)$, which tells us the probability that the precipitation is $\leq x$ on any given day. When x is continuous and has a continuous distribution, $F(x)$ should be uniformly distributed.[10] The CDF of a two-cluster mixture is

$$F(x) = \lambda_1 F_1(x) + \lambda_2 F_2(x) \tag{17.32}$$

and similarly for more clusters. A little R experimentation gives a function for computing the CDF of a Gaussian mixture:

```
pnormmix <- function(x,mixture) {
  lambda <- mixture$lambda
  k <- length(lambda)
  pnorm.from.mix <- function(x,cluster) {
    lambda[cluster]*pnorm(x,mean=mixture$mu[cluster],
                          sd=mixture$sigma[cluster])
  }
  pnorms <- sapply(1:k,pnorm.from.mix,x=x)
  return(rowSums(pnorms))
}
```

[10] We saw this principle when we looked at generating random variables in Chapter 5.




```
plot(hist(snoq,breaks=101),col="grey",border="grey",freq=FALSE,
     xlab="Precipitation (1/100 inch)",main="Precipitation in Snoqualmie Falls")
lines(density(snoq),lty=2)
invisible(sapply(1:2,plot.gaussian.clusters,mixture=snoq.k2))
```

Figure 17.3 As in the previous figure, plus the clusters of a mixture of two Gaussians, fitted to the data by the EM algorithm (dashed lines). These are scaled by the mixing weights of the clusters. Could you add the *sum* of the two cluster densities to the plot?

We can use this to get a plot like Figure 17.4. We do not have the tools to

assess whether the *size* of the departure from the main diagonal is significant[11], but the fact that the errors are so very structured is rather suspicious.

17.4.4 Selecting the Number of Clusters by Cross-Validation

Since a two-cluster mixture seems iffy, we could consider using more clusters. By going to three, four, etc., clusters, we improve our in-sample likelihood, but of course expose ourselves to the danger of over-fitting. Some sort of model selection is called for. We could do cross-validation, or we could do hypothesis testing. Let's try cross-validation first.

We can already do fitting, but we need to calculate the log-likelihood on the held-out data. As usual, let's write a function; in fact, let's write two.

```
# Probability density corresponding to a Gaussian mixture model
# Inputs: location for evaluating the pdf (x)
  # mixture-model object (mixture)
  # whether or not output should be logged (log)
# Output: the (possibly logged) PDF at the point(s) x
dnormalmix <- function(x,mixture,log=FALSE) {
  lambda <- mixture$lambda
  k <- length(lambda)
  # Calculate share of likelihood for all data for one cluster
  like.cluster <- function(x,cluster) {
    lambda[cluster]*dnorm(x,mean=mixture$mu[cluster],
                          sd=mixture$sigma[cluster])
  }
  # Create array with likelihood shares from all clusters over all data
  likes <- sapply(1:k,like.cluster,x=x)
  # Add up contributions from clusters
  d <- rowSums(likes)
  if (log) {
    d <- log(d)
  }
  return(d)
}

# Evaluate the loglikelihood of a mixture model at a vector of points
# Inputs: vector of data points (x)
  # mixture model object (mixture)
# Output: sum of log probability densities over the points in x
loglike.normalmix <- function(x,mixture) {
  loglike <- dnormalmix(x,mixture,log=TRUE)
  return(sum(loglike))
}
```

To check that we haven't made a big mistake in the coding:

```
loglike.normalmix(snoq,mixture=snoq.k2)
## [1] -32681.21
```

which matches the log-likelihood reported by `summary(snoq.k2)`. But our function can be used on different data!

[11] Though we could: the most straight-forward thing to do would be to simulate from the mixture, and repeat this with simulation output.

We *could* do five-fold or ten-fold CV, but just to illustrate the approach we'll do simple data-set splitting, where a randomly-selected half of the data is used to fit the model, and half to test.

```
n <- length(snoq)
data.points <- 1:n
data.points <- sample(data.points) # Permute randomly
train <- data.points[1:floor(n/2)] # First random half is training
test <- data.points[-(1:floor(n/2))] # 2nd random half is testing
candidate.cluster.numbers <- 2:10
loglikes <- vector(length=1+length(candidate.cluster.numbers))
# k=1 needs special handling
mu<-mean(snoq[train]) # MLE of mean
sigma <- sd(snoq[train])*sqrt((n-1)/n) # MLE of standard deviation
loglikes[1] <- sum(dnorm(snoq[test],mu,sigma,log=TRUE))
for (k in candidate.cluster.numbers) {
  mixture <- normalmixEM(snoq[train],k=k,maxit=400,epsilon=1e-2)
  loglikes[k] <- loglike.normalmix(snoq[test],mixture=mixture)
}
```

When you run this, you will may see a lot of warning messages saying "One of the variances is going to zero; trying new starting values." The issue is that we can give any one value of x arbitrarily high likelihood by centering a Gaussian there and letting its variance shrink towards zero. This is however generally considered unhelpful — it leads towards the pathologies that keep us from doing pure maximum likelihood estimation in non-parametric problems (Chapter 14) — so when that happens the code recognizes it and starts over.

If we look at the log-likelihoods, we see that there is a dramatic improvement with the first few clusters, and then things slow down a lot[12]:

```
loglikes
##  [1] -17605.68 -16373.96 -15756.05 -15526.01 -15368.76 -15303.89 -15252.24
##  [8] -15245.16 -15239.98 -15234.76
```

(See also Figure 17.5). This favors nine clusters to the mixture. It looks like Figure 17.6. The calibration is now nearly perfect, at least on the training data (Figure 17.7).

17.4.5 Interpreting the Clusters in the Mixture, or Not

The clusters of the mixture are far from arbitrary. It appears from Figure 17.6 that as the mean increases, so does the variance. This impression is confirmed from Figure 17.8. Now it *could* be that there really are nine types of rainy days in Snoqualmie Falls which just so happen to have this pattern of distributions, but this seems a bit suspicious — as though the mixture is trying to use Gaussians systematically to approximate a fundamentally different distribution, rather

[12] Notice that the numbers here are about half of the log-likelihood we calculated for the two-cluster mixture on the complete data. This is as it should be, because log-likelihood is proportional to the number of observations. (Why?) It's more like the *sum* of squared errors than the *mean* squared error. If we want something which is directly comparable across data sets of different size, we should use the log-likelihood per observation.

than get at something which really is composed of nine distinct Gaussians. This judgment relies on our scientific understanding of the weather, which makes us surprised by seeing a pattern like this in the parameters. (Calling this "scientific knowledge" is a bit excessive, but you get the idea.) Of course we are sometimes wrong about things like this, so it is certainly not conclusive. Maybe there really *are* nine types of days, each with a Gaussian distribution, and some subtle meteorological reason why their means and variances should be linked like this. For that matter, maybe this is a sign that the meteorologists have missed something and should work to *discover* nine distinct types of days.

There are two directions to take this: the purely statistical one, and the substantive one.

On the purely statistical side, if all we care about is being able to describe the distribution of the data and to predict future precipitation, then it doesn't really matter whether the nine-cluster Gaussian mixture is true in any ultimate sense. Cross-validation picked nine clusters not because there really are nine types of days, but because a nine-cluster model had the best trade-off between approximation bias and estimation variance. The selected mixture gives a pretty good account of itself, nearly the same as the kernel density estimate (Figure 17.9). It requires 26 parameters[13], which may seem like a lot, but the kernel density estimate requires keeping around all 6920 data points plus a bandwidth. On sheer economy, the mixture then has a lot to recommend it.

On the substantive side, there are various things we could do to check the idea that wet days really do divide into nine types. These are going to be informed by our background knowledge about the weather. One of the things we know, for example, is that weather patterns more or less repeat in an annual cycle, and that different types of weather are more common in some parts of the year than in others. If, for example, we consistently find type 6 days in August, that suggests that is at least compatible with these being real, meteorological patterns, and not just approximation artifacts.

Let's try to look into this visually. `snoq.k9$posterior` is a 6920×9 array which gives the probability for each day to belong to each class. I'll boil this down to assigning each day to its most probable class:

```
day.classes <- apply(snoq.k9$posterior,1,which.max)
```

We can't just plot this and hope to see any useful patterns, because we want to see stuff recurring every year, and we've stripped out the dry days, the division into years, the padding to handle leap-days, etc. Thus, we need to do a bit of R magic. Remember we started with a giant vector `snoqualmie` which had all days, wet or dry; let's copy that into a data frame, to which we'll add the classes and the days of the year.

```
snoqualmie.classes <- data.frame(precip=snoqualmie, class=0)
years <- 1948:1983
```

[13] A mean and a standard deviation for each of nine clusters (=18 parameters), plus mixing weights (nine of them, but they have to add up to one).

```
snoqualmie.classes$day <- rep(c(1:366,1:365,1:365,1:365),times=length(years)/4)
wet.days <- (snoqualmie > 0)
snoqualmie.classes$class[wet.days] <- day.classes
```

Now, it's somewhat inconvenient that the index numbers of the clusters do not really tell us about the mean amount of precipitation. Let's try replacing the numerical labels in `snoqualmie.classes` by those means.

```
snoqualmie.classes$class[wet.days] <- snoq.k9$mu[day.classes]
```

This leaves alone dry days (still zero) and NA days (still NA). Now we can plot (Figure 17.10).

The result is discouraging if we want to read any deeper meaning into the classes. The class with the heaviest amounts of precipitation is most common in the winter, but so is the classes with the second-heaviest amount of precipitation, the etc. It looks like the weather changes smoothly, rather than really having discrete classes. In this case, the mixture model seems to be merely a predictive device, and not a revelation of hidden structure.[14]

17.4.6 Hypothesis Testing for Mixture-Model Selection

An alternative to using cross-validation to select the number of mixtures is to use hypothesis testing. The k-cluster Gaussian mixture model is nested within the $(k+1)$-cluster model, so the latter must have a strictly higher likelihood on the training data. If the data really comes from a k-cluster mixture (the null hypothesis), then this extra increment of likelihood will follow one distribution, but if the data come from a larger model (the alternative), the distribution will be different, and stochastically larger.

Based on general likelihood theory [[CROSS-REF]], we might expect that the null distribution is, for large sample sizes,

$$2(\log L_{k+1} - \log L_k) \sim \chi^2_{dim(k+1)-dim(k)} \tag{17.33}$$

where L_k is the likelihood under the k-cluster mixture model, and $dim(k)$ is the number of parameters in that model. There are however several reasons to distrust such an approximation, including the fact that we are approximating the likelihood through the EM algorithm. We can instead just find the null distribution by simulating from the smaller model, which is to say we can do a parametric bootstrap.

[14] A a distribution called a "type II generalized Pareto", where $p(x) \propto (1 + x/\sigma)^{-\theta-1}$, provides a decent fit here. (See Shalizi 2007; Arnold 1983 on this distribution and its estimation.) With only two parameters, rather than 26, its log-likelihood is only 1% higher than that of the nine-cluster mixture, and it is almost but not quite as calibrated. One origin of the type II Pareto is as a mixture of exponentials (Maguire *et al.*, 1952). If $X|Z \sim \text{Exp}(\sigma/Z)$, and Z itself has a Gamma distribution, $Z \sim \Gamma(\theta, 1)$, then the unconditional distribution of X is type II Pareto with scale σ and shape θ. We might therefore investigate fitting a finite mixture of exponentials, rather than of Gaussians, for the Snoqualmie Falls data. We might of course still end up concluding that there is a continuum of different sorts of days, rather than a finite set of discrete types.

While it is not too hard to program this by hand (Exercise 17.7), the `mixtools` package contains a function to do this for us, called `boot.comp`, for "bootstrap comparison". Let's try it out (Figure 17.11).

The command in the figure tells `boot.comp` to consider mixtures of up to 10 clusters (just as we did with cross-validation), increasing the size of the mixture it uses when the difference between k and $k+1$ is significant. (The default is "significant at the 5% level", as assessed by 100 bootstrap replicates, but that's controllable.) The command also tells it what kind of mixture to use, and passes along control settings to the EM algorithm which does the fitting. Each individual fit is fairly time-consuming, and we are requiring 200 at each value of k. This took about three minutes to run on my laptop.

This selected three clusters (rather than nine), and accompanied this decision with a rather nice trio of histograms explaining why (Figure 17.11). Remember that `boot.comp` stops expanding the model when there's even a 5% chance of that the apparent improvement could be due to mere over-fitting. This is actually pretty conservative, and so ends up with rather fewer clusters than cross-validation.

Let's explore the output of `boot.comp`, conveniently stored in the object `snoq.boot`.

```
str(snoq.boot)
## List of 3
##  $ p.values   : num [1:4] 0 0 0.03 0.28
##  $ log.lik    :List of 4
##   ..$ : num [1:100] 3.11 7.4 4.4 2.22 4.12 ...
##   ..$ : num [1:100] 3.4926 2.33211 3.5407 0.00455 2.34426 ...
##   ..$ : num [1:100] 3.97 5.46 2432.9 2.5 2.89 ...
##   ..$ : num [1:100] 0.012 1.591 1.543 0.414 0.114 ...
##  $ obs.log.lik: num [1:4] 5096 2354 920 562
```

This tells us that `snoq.boot` is a list with three elements, called `p.values`, `log.lik` and `obs.log.lik`, and tells us a bit about each of them. `p.values` contains the p-values for testing H_1 (one cluster) against H_2 (two clusters), testing H_2 against H_3, and H_3 against H_4. Since we set a threshold p-value of 0.05, it stopped at the last test, accepting H_3. (Under these circumstances, if the difference between $k = 3$ and $k = 4$ was really important to us, it would probably be wise to increase the number of bootstrap replicates, to get more accurate p-values.) `log.lik` is itself a list containing the bootstrapped log-likelihood ratios for the three hypothesis tests; `obs.log.lik` is the vector of corresponding observed values of the test statistic.

Looking back to Figure 17.5, there is indeed a dramatic improvement in the generalization ability of the model going from one cluster to two, and from two to three, and diminishing returns to complexity thereafter. Stopping at $k = 3$ produces pretty reasonable results, though repeating the exercise of Figure 17.10 is no more encouraging for the reality of the latent classes.

17.5 Further Reading

My presentation of the EM algorithm draws *heavily* on Neal and Hinton (1998).

The EM algorithm is so useful and general that it is applied to *lots* of problems with missing data or latent variables, and has in fact been re-invented multiple times. (For instance, some old methods of estimating factor models were basically the EM algorithm.) The name "EM algorithm" comes from the statistics of mixture models in the late 1970s.

A common problem in time-series analysis and signal processing is that of "filtering" or "state estimation": there's an unknown signal S_t, which we want to know, but all we get to observe is some noisy, corrupted measurement, $X_t = h(S_t) + \eta_t$. (A historically important example of a "state" to be estimated from noisy measurements is "Where is our rocket and which way is it headed?" — see McGee and Schmidt, 1985.) This is solved by the EM algorithm, with the signal as the hidden variable; Fraser (2008) gives a really good introduction to such models and how they use EM. Since the 1960s the EM algorithm in this context has been known as the "Baum-Welch" algorithm.

Instead of just doing mixtures of densities, one can also do mixtures of predictive models, say mixtures of regressions, or mixtures of classifiers. The hidden variable Z here controls which regression function to use. A general form of this is what's known as a **mixture-of-experts** model (Jordan and Jacobs, 1994; Jacobs, 1997) — each predictive model is an "expert", and there can be a quite complicated set of hidden variables determining which expert to use when.

Exercises

17.1 Write a function to simulate from a Gaussian mixture model. Check that it works by comparing a density estimated on its output to the theoretical density.

17.2 Show that the mixture of a $\text{Binom}(2, 0.75)$ and a $\text{Binom}(2, 0.25)$ is *not* a $\text{Binom}(2, p)$ for any p

17.3 Following §17.2, suppose that we want to estimate the λ_j by maximizing the likelihood.

1. Show that
$$\frac{\partial \ell}{\partial \lambda_j} = \sum_{i=1}^{n} w_{ij} \tag{17.34}$$
2. Explain why we need to add a Lagrange multiplier to enforce the constraint $\sum_{j=1}^{K} \lambda_j = 1$, and why it was OK to ignore that in Eq. 17.10.
3. Show that, including the Lagrange multiplier, the optimal value of λ_j is $\sum_{i=1}^{n} w_{ij}/n$. Can you find a simple expression for the Lagrange multiplier?

17.4 Work through the E- step and M- step for a mixture of two Poisson distributions.

17.5 Code up the EM algorithm for a mixture of K Gaussians. Simulate data from $K = 3$ Gaussians. How well does your code assign data-points to clusters if you give it the actual Gaussian parameters as your initial guess? If you give it other initial parameters?

17.6 Prove Eq. 17.18.

17.7 Write a function to find the distribution of the log-likelihood ratio for testing the hypothesis that the mixture has k Gaussian clusters against the alternative that it has $k + 1$, by

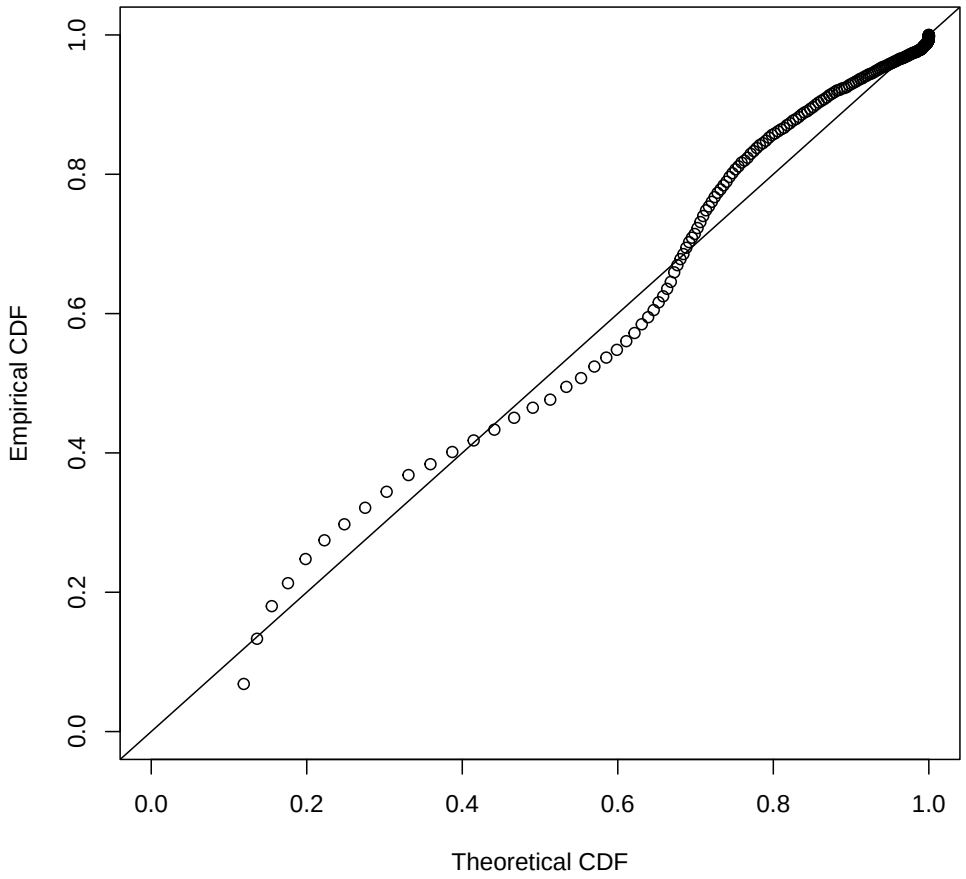
simulating from the k-cluster model. Compare the output to the `boot.comp` function in `mixtools`.

17.8 Write a function to fit a mixture of exponential distributions using the EM algorithm. Does it do any better at discovering sensible structure in the Snoqualmie Falls data?

17.9 Explain how to use relative distribution plots (Chapter F) to check calibration, along the lines of Figure 17.4.

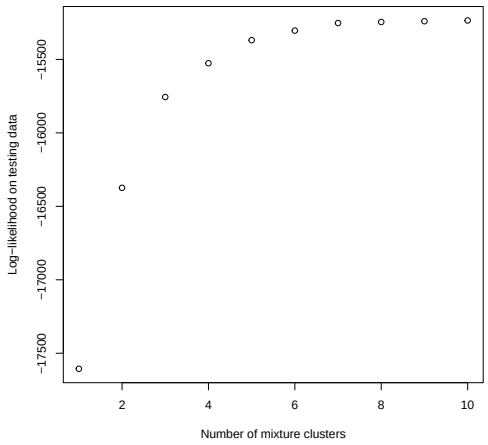
17.10 *Abstract vs. concrete* The abstract EM algorithm of §17.2.1 is very general, much more general than the concrete algorithm given on the previous pages. Nonetheless, the former reduces to the latter when the latent variable Z follows a multinomial distribution.

1. Show that the M step of the abstract EM algorithm is equivalent to solving
$$\sum_{i=1}^{n} w_{ij} \frac{\partial \log f(x_i; \theta_j)}{\partial \theta_j} = 0 \tag{17.35}$$
for the new θ.
2. Show that the maximization in the E step of the abstract EM algorithm yields Eq. 17.13.



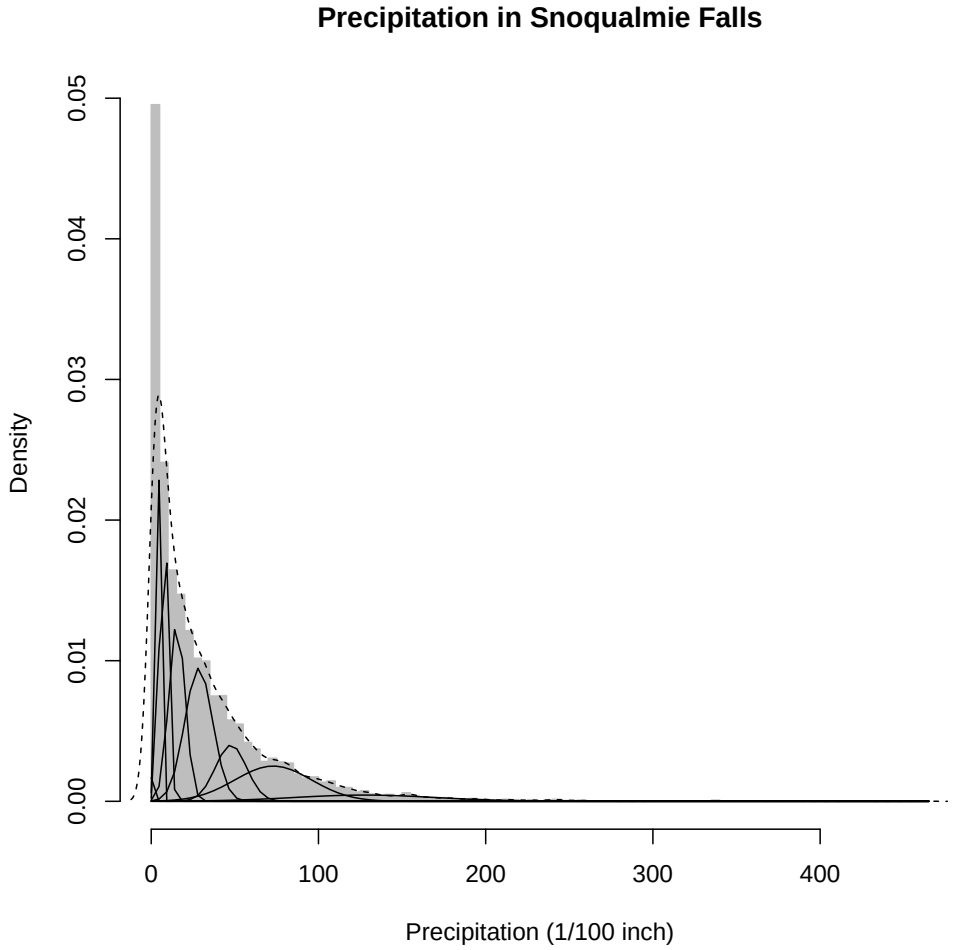

```
distinct.snoq <- sort(unique(snoq))
tcdfs <- pnormmix(distinct.snoq,mixture=snoq.k2)
ecdfs <- ecdf(snoq)(distinct.snoq)
plot(tcdfs,ecdfs,xlab="Theoretical CDF",ylab="Empirical CDF",xlim=c(0,1),
     ylim=c(0,1))
abline(0,1)
```

Figure 17.4 Calibration plot for the two-cluster Gaussian mixture. For each distinct value of precipitation x, we plot the fraction of days predicted by the mixture model to have $\leq x$ precipitation on the horizontal axis, versus the actual fraction of days $\leq x$.



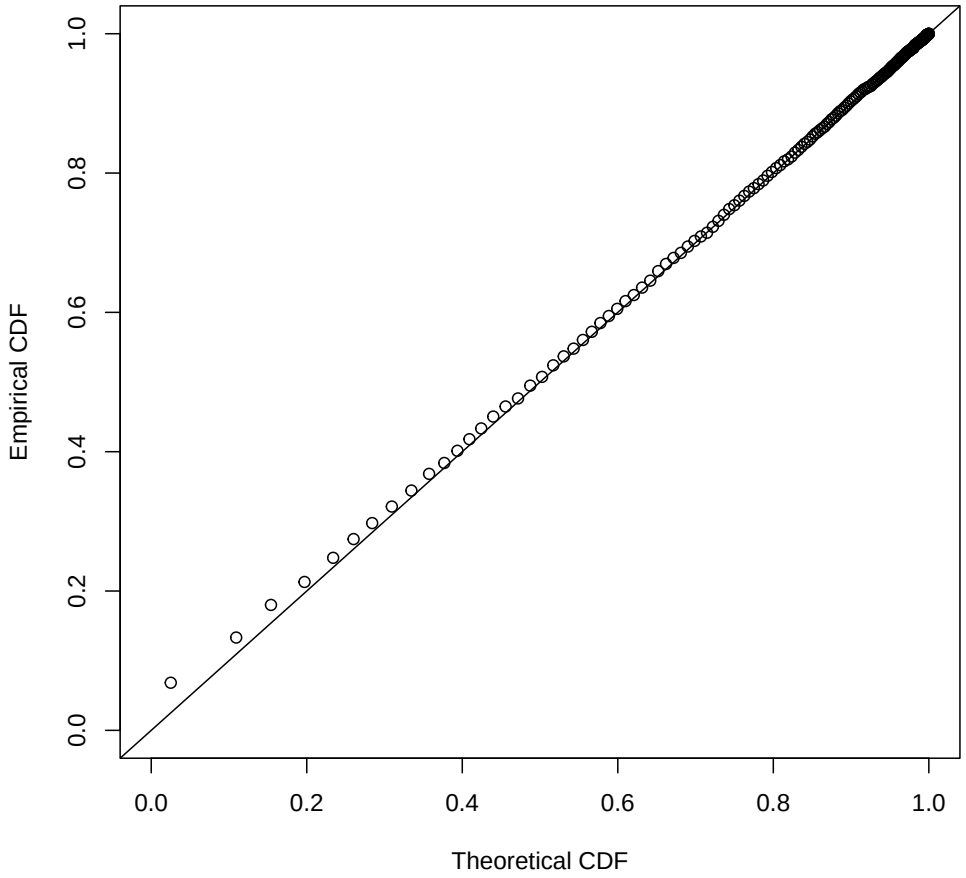

```
plot(x=1:10, y=loglikes,xlab="Number of mixture clusters",
     ylab="Log-likelihood on testing data")
```

Figure 17.5 Log-likelihoods of different sizes of mixture models, fit to a random half of the data for training, and evaluated on the other half of the data for testing.



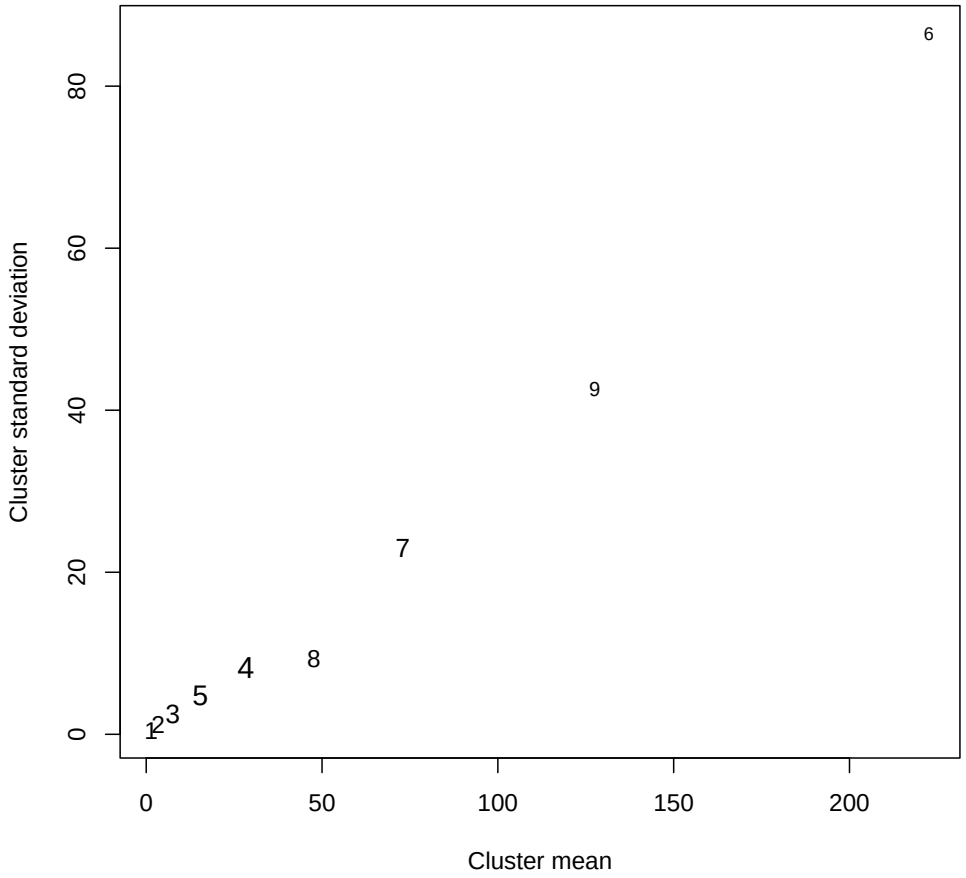

```
snoq.k9 <- normalmixEM(snoq,k=9,maxit=400,epsilon=1e-2)
plot(hist(snoq,breaks=101),col="grey",border="grey",freq=FALSE,
     xlab="Precipitation (1/100 inch)",main="Precipitation in Snoqualmie Falls")
lines(density(snoq),lty=2)
invisible(sapply(1:9,plot.gaussian.clusters,mixture=snoq.k9))
```

Figure 17.6 As in Figure 17.3, but using the nine-cluster Gaussian mixture.



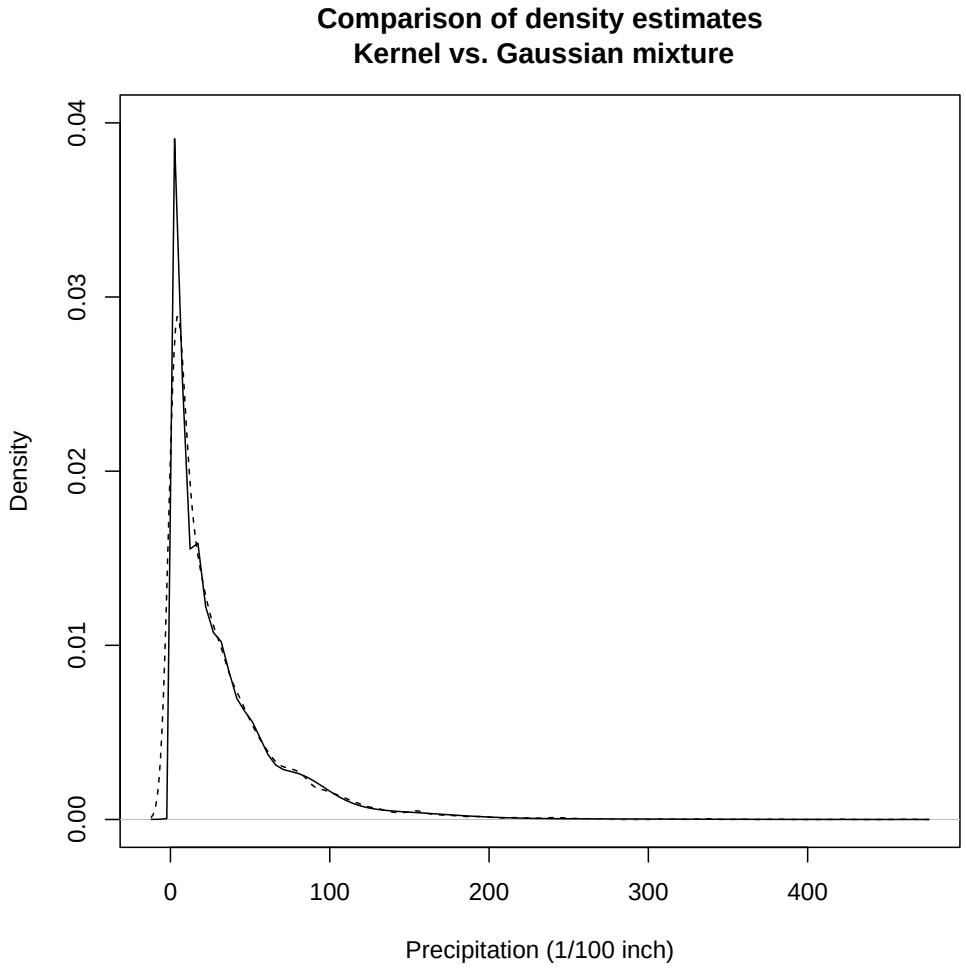

```
distinct.snoq <- sort(unique(snoq))
tcdfs <- pnormmix(distinct.snoq,mixture=snoq.k9)
ecdfs <- ecdf(snoq)(distinct.snoq)
plot(tcdfs,ecdfs,xlab="Theoretical CDF",ylab="Empirical CDF",xlim=c(0,1),
     ylim=c(0,1))
abline(0,1)
```

Figure 17.7 Calibration plot for the nine-cluster Gaussian mixture.



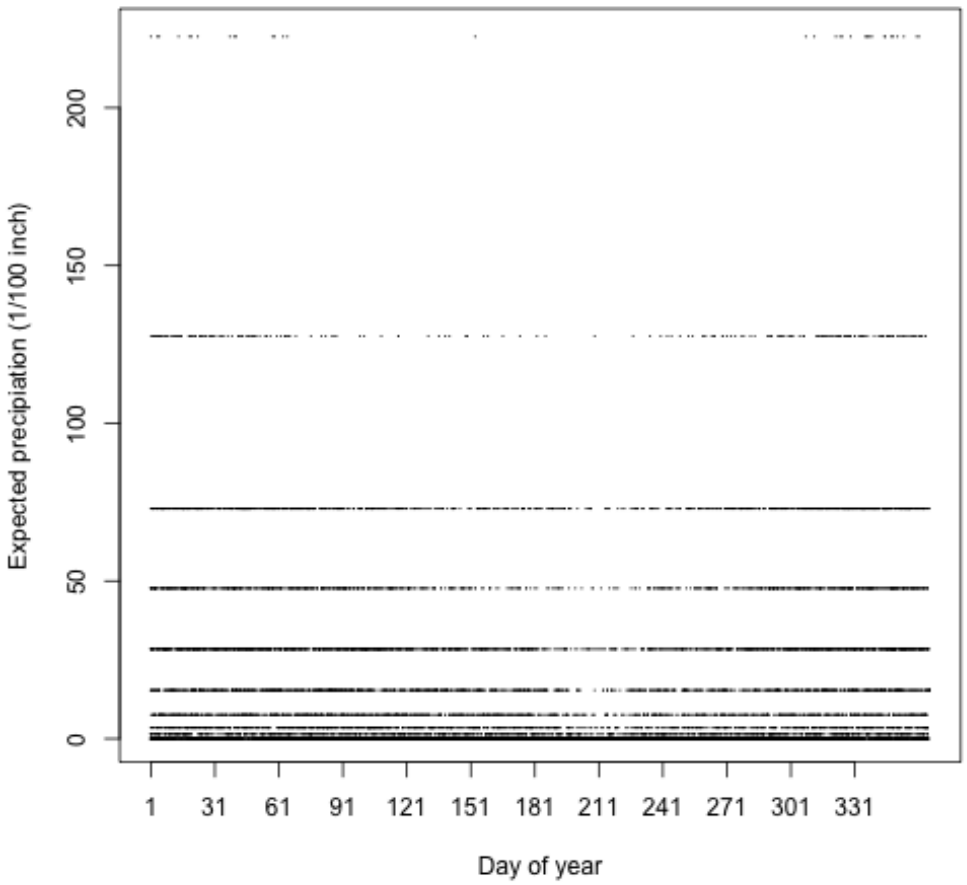

```
plot(0,xlim=range(snoq.k9$mu),ylim=range(snoq.k9$sigma),type="n",
     xlab="Cluster mean", ylab="Cluster standard deviation")
points(x=snoq.k9$mu,y=snoq.k9$sigma,pch=as.character(1:9),
       cex=sqrt(0.5+5*snoq.k9$lambda))
```

Figure 17.8 Characteristics of the clusters of the 9-mode Gaussian mixture. The horizontal axis gives the cluster mean, the vertical axis its standard deviation. The area of the number representing each cluster is proportional to the cluster's mixing weight.



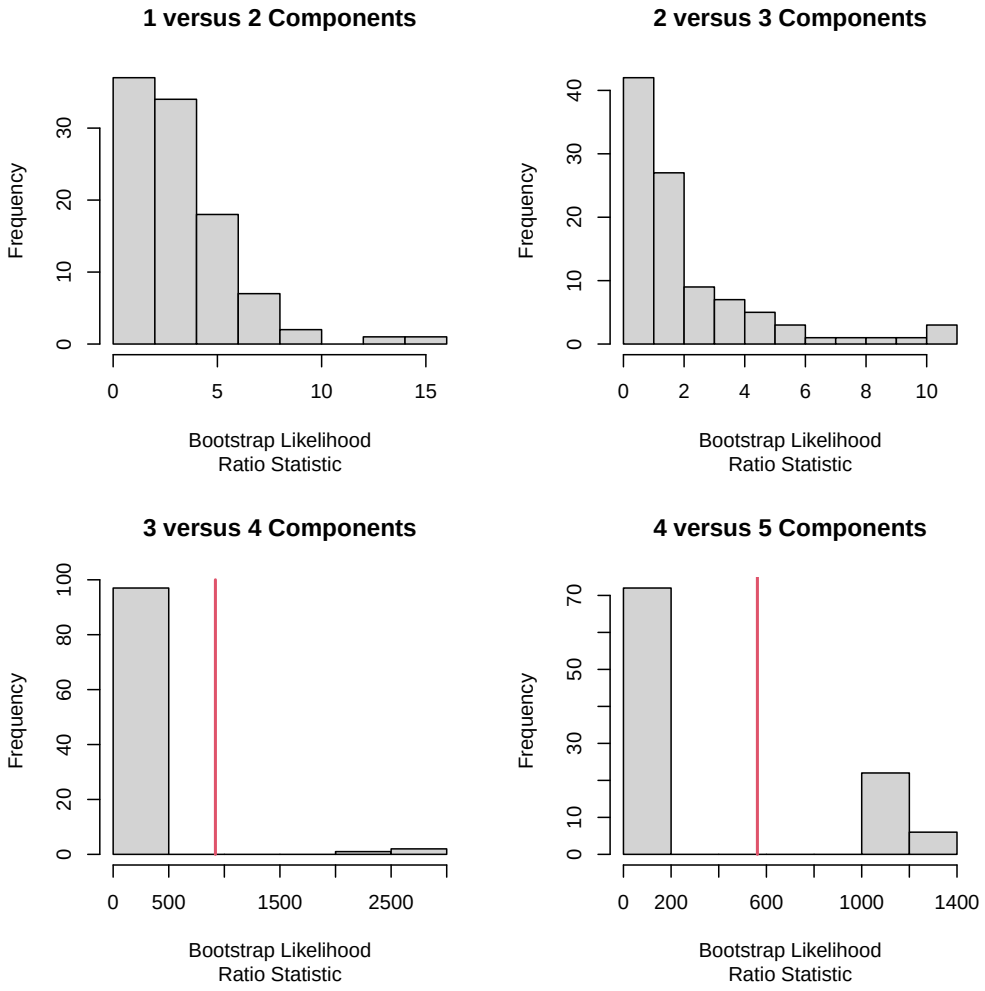

```
plot(density(snoq),lty=2,ylim=c(0,0.04),
     main=paste("Comparison of density estimates\n",
                "Kernel vs. Gaussian mixture"),
     xlab="Precipitation (1/100 inch)")
curve(dnormalmix(x,snoq.k9),add=TRUE)
```

Figure 17.9 Dashed line: kernel density estimate. Solid line: the nine-Gaussian mixture. Notice that the mixture, unlike the KDE, gives negligible probability to negative precipitation.




```
plot(x=snoqualmie.classes$day, y=snoqualmie.classes$class,
     xlim=c(1,366),ylim=range(snoq.k9$mu),xaxt="n",
     xlab="Day of year",ylab="Expected precipiation (1/100 inch)",
     pch=16,cex=0.2)
axis(1,at=1+(0:11)*30)
```

Figure 17.10 Plot of days classified according to the nine-cluster mixture. Horizontal axis: day of the year, numbered from 1 to 366 (to handle leap-years). Vertical axis: expected amount of precipitation on that day, according to the most probable class for the day.




```
snoq.boot <- boot.comp(snoq,max.comp=10,mix.type="normalmix",
                       maxit=400,epsilon=1e-2)
```

Figure 17.11 Histograms produced by `boot.comp()`. The vertical red lines mark the observed difference in log-likelihoods.

18

Graphical Models

We have spent a lot of time looking at ways of figuring out how one variable (or set of variables) depends on another variable (or set of variables) — this is the core idea in regression and in conditional density estimation. We have also looked at how to estimate the joint distribution of variables, both with kernel density estimation and with models like factor and mixture models. The later two show an example of how to get the joint distribution by combining a conditional distribution (observables given factors; mixture components) with a marginal distribution (Gaussian distribution of factors; the component weights). When dealing with complex sets of dependent variables, it would be nice to have a general way of composing conditional distributions together to get joint distributions, and especially nice if this gave us a way of reasoning about what we could ignore, of seeing which variables are irrelevant to which other variables. This is what **graphical models** let us do.

18.1 Conditional Independence and Factor Models

The easiest way into this may be to start with the diagrams we drew for factor analysis. There, we had observables and we had factors, and each observable depended on, or loaded on, some of the factors. We drew a diagram where we had nodes, standing for the variables, and arrows running from the factors to the observables which depended on them. In the factor model, all the observables were conditionally independent of each other, given all the factors:

$$p(X_1, X_2, \ldots X_p | F_1, F_2, \ldots F_q) = \prod_{i=1}^{p} p(X_i | F_1, \ldots F_q) \tag{18.1}$$

But in fact observables are also independent of the factors they do not load on, so this is still too complicated. Let's write $\text{loads}(i)$ for the set of factors on which the observable X_i loads. Then

$$p(X_1, X_2, \ldots X_p | F_1, F_2, \ldots F_q) = \prod_{i=1}^{p} p(X_i | F_{\text{loads}(i)}) \tag{18.2}$$

Consider Figure 18.1. The conditional distribution of observables given factors is

$$p(X_1, X_2, X_3, X_4 | F_1, F_2) = p(X_1 | F_1, F_2) p(X_2 | F_1, F_2) p(X_3 | F_1) p(X_4 | F2) \tag{18.3}$$

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

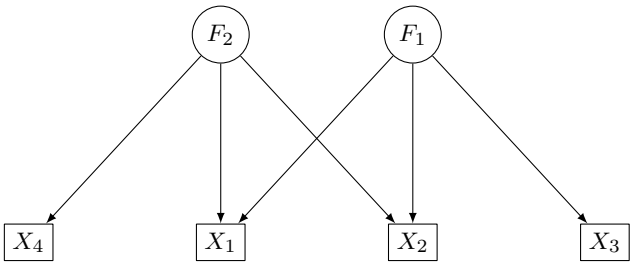


Figure 18.1 Illustration of a typical model with two latent factors (F_1 and F_2, in circles) and four observables (X_1 through X_4).

X_1 loads on F_1 and F_2, so it is independent of everything else, given those two variables. X_1 is unconditionally dependent on X_2, because they load on common factors, F_1 and F_2; and X_1 and X_3 are also dependent, because they both load on F_1. In fact, X_1 and X_2 are still dependent given F_1, because X_2 still gives information about F_2. But X_1 and X_3 are independent given F_1, because they have no other factors in common. Finally, X_3 and X_4 are unconditionally independent because they have no factors in common. But they become dependent given X_1, which provides information about both the common factors.

None of these assertions rely on the detailed assumptions of the factor model, like Gaussian distributions for the factors, or linear dependence between factors and observables. What they rely on is that X_i is independent of *everything else*, given the factors it loads on. The idea of graphical models is to generalize this, by focusing on relations of direct dependence, and the conditional independence relations implied by them.

18.2 Directed Acyclic Graph (DAG) Models

We have a collection of variables, which to be generic I'll write $X_1, X_2, \ldots X_p$. These may be discrete, continuous, or even vectors; it doesn't matter. We represent these visually as nodes in a graph. There are arrows connecting some of these nodes. If an arrow runs from X_i to X_j, then X_i is a **parent** of X_j. This is, as the name "parent" suggests, an anti-symmetric relationship, i.e., X_j cannot also be the parent of X_i. This is why we use an arrow, and why the graph is **directed**[1]. We write the set of all parents of X_j as parents(j); this generalizes the notion of the factors which an observable loads on to. The joint distribution "decomposes according to the graph" or "factors according to the graph":

$$p(X_1, X_2, \ldots X_p) = \prod_{i=1}^{p} p(X_i | X_{\text{parents}(i)}) \tag{18.4}$$

If X_i has no parents, because it has no incoming arrows, take $p(X_i|X_{\text{parents}(i)})$ just to be the marginal distribution $p(X_i)$. Such variables are called **exogenous**;

[1] See Appendix H for a brief review of the ideas and jargon of graph theory.

the others, with parents, are **endogenous**. An unfortunate situation could arise where X_1 is the parent of X_2, which is the parent of X_3, which is the parent of X_1. Perhaps, under some circumstances, we could make sense of this and actually calculate with Eq. 18.4, but the general practice is to rule it out by assuming the graph is **acyclic**, i.e., that it has no cycles, i.e., that we cannot, by following a series of arrows in the graph, go from one node to other nodes and ultimately back to our starting point. Altogether we say that we have a **directed acyclic graph**, or **DAG**, which represents the direct dependencies between variables.[2]

What good is this? The primary virtue is that if we are dealing with a DAG model, the graph tells us all the dependencies we need to know; those are the conditional distributions of variables on their parents, appearing in the product on the right hand side of Eq. 18.4. (This includes the distribution of the exogeneous variables.) This fact has two powerful sets of implications, for probabilistic reasoning and for statistical inference.

Let's take inference first, because it's more obvious: all that we have to estimate are the conditional distributions $p(X_i|X_{\text{parents}(i)})$. We do not have to estimate the distribution of X_i given *all* of the other variables, unless of course they are all parents of X_i. Since estimating distributions, or even just regressions, conditional on many variables is hard, it is extremely helpful to be able to read off from the graph which variables we can *ignore*. Indeed, if the graph tells us that X_i is exogeneous, we don't have to estimate it conditional on anything, we just have to estimate its marginal distribution.

18.2.1 Conditional Independence and the Markov Property

The probabilistic implication of Eq. 18.4 is perhaps even more important, and that has to do with conditional independence. Pick any two variables X_i and X_j, where X_j is not a parent of X_i. Consider the distribution of X_i conditional on its parents *and* X_j. There are two possibilities. (i) X_j is not a descendant of X_i. Then we can see that X_i and X_j are conditionally independent. This is true *no matter what* the actual conditional distribution functions involved are; it's just implied by the joint distribution respecting the graph. (ii) Alternatively, X_j is a descendant of X_i. Then in general they are not independent, even conditional on the parents of X_i. So the graph implies that certain conditional independence relations will hold, but that others in general will *not* hold.

As you know from your probability courses, a sequence of random variables $X_1, X_2, X_3, \ldots$ forms a **Markov process**[3] when "the past is independent of the future given the present": that is,

$$X_{t+1} \perp\!\!\!\perp (X_{t-1}, X_{t-2}, \ldots X_1)|X_t \tag{18.5}$$

[2] See §18.6 for remarks on undirected graphical models, and graphs with cycles.

[3] After the Russian mathematician A. A. Markov, who introduced the theory of Markov processes in the course of a mathematical dispute with his arch-nemesis, to show that probability and statistics could apply to dependent events, and hence that Christianity was not *necessarily* true (I am not making this up: Basharin *et al.*, 2004).

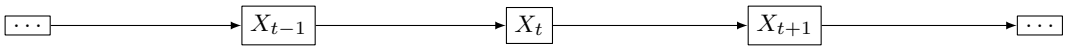


Figure 18.2 DAG for a discrete-time Markov process. At each time t, X_t is the child of X_{t-1} alone, and in turn the sole parent of X_{t+1}.

from which it follows that

$$(X_{t+1}, X_{t+2}, X_{t+3}, \ldots) \perp\!\!\!\perp (X_{t-1}, X_{t-2}, \ldots X_1) | X_t \tag{18.6}$$

which is called the **Markov property**. DAG models have a similar property: if we take any collection of nodes I, it is independent of its non-descendants, given its parents:

$$X_I \perp\!\!\!\perp X_{\text{non-descendants}(I)} | X_{\text{parents}(I)} \tag{18.7}$$

This is the **directed graph Markov property**. The ordinary Markov property is a special case, when the graph looks like Figure 18.2[4].

On the other hand, if we condition on one of X_i's children, X_i will generally be dependent on any other parent of that child. If we condition on multiple children of X_i, we'll generally find X_i is dependent on all its co-parents. It should be plausible, and is in fact true, that X_i is independent of everything else in the graph if we condition on its parents, its children, and its children's other parents. This set of nodes is called X_i's **Markov blanket**.

18.3 Conditional Independence and D-Separation

It is clearly very important to us to be able to deduce when two sets of variables are conditionally independent of each other given a third. One of the great uses of DAGs is that they give us a fairly simple criterion for this, in terms of the graph itself. *All* distributions which conform to a given DAG share a common set of conditional independence relations, implied by the Markov property, no matter what their parameters or the form of the distributions.

Our starting point is that when we have a single directed edge, we can reason from the parent to the child, or from the child to the parent. While (as we'll see in Part III) it's reasonable to say that *influence* or *causation* flows one way, along the direction of the arrows, *statistical information* can flow in either direction. Since dependence is the presence of such statistical information, if we want to figure out which variables are dependent on which, we need to keep track of these information flows.

While we can do inference in either direction across any one edge, we may or may not be able to propagate this information further. Consider the four graphs in Figure 18.3. In every case, we condition on X, which acts as the source of information. In the first three cases, we can (in general) propagate the information

[4] To see this, take the "future" nodes, indexed by $t+1$ and up, as the set I. Their parent consists just of X_t, and all their non-descendants are the even earlier nodes at times $t-1$, $t-2$, etc.

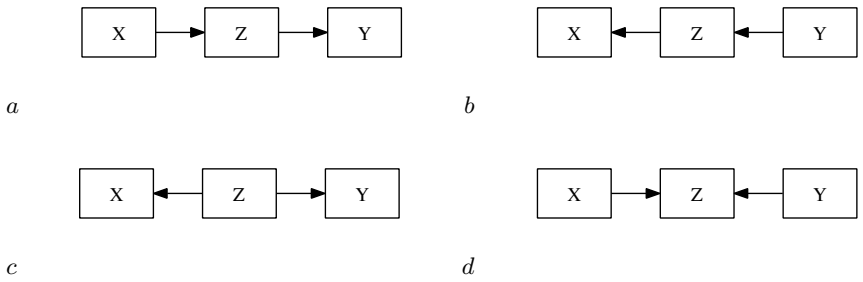


Figure 18.3 Four DAGs for three linked variables. The first two (a and b) are called **chains**; c is a **fork**; d is a **collider**. If these were the whole of the graph, we would have $X \not\perp\!\!\!\perp Y$ and $X \perp\!\!\!\perp Y|Z$. For the collider, however, we would have $X \perp\!\!\!\perp Y$ while $X \not\perp\!\!\!\perp Y|Z$.

from X to Z to Y — the Markov property tells us that Y is independent of its non-descendants given its parents, but in none of those cases does that make X and Y independent. In the last graph, however, what's called a **collider**[5], we cannot propagate the information, because Y has no parents, and X is not its descendant, hence they are independent. We learn about Z from X, but this doesn't tell us anything about Z's other cause, Y.

All of this flips around when we condition on the intermediate variable (Z in Figure 18.3). In the **chains** (Figures 18.3a and b), conditioning on the intermediate variable blocks the flow of information from X to Y — we learn nothing more about Y from X and Z than from Z alone, at least not along this path. This is also true of the **fork** (Figure 18.3c) — conditional on their common cause, the two effects are uninformative about each other. But in a collider, conditioning on the common effect Z makes X and Y dependent on each other, as we've seen before. In fact, if we don't condition on Z, but do condition on a descendant of Z, we also create dependence between Z's parents.

We are now in a position to work out conditional independence relations. We pick our two favorite variables, X and Y, and condition them both on some third set of variables S. If S **blocks** every undirected path[6] from X to Y, then they must be conditionally independent given S. An unblocked path is also called **active**. A path is active when every variable along the path is active; if even one variable is blocked by S, the whole path is blocked. A variable Z along a path is active, conditioning on S, if

1. Z is a collider along the path, and in S; or,
2. Z is a descendant of a collider, and in S; or
3. Z is not a collider, and not in S.

Turned around, Z is blocked or de-activated by conditioning on S if

[5] Because two incoming arrows "collide" there.
[6] Whenever I talk about undirected paths, I mean paths without cycles.

1. Z is a non-collider and in S; or
2. Z is collider, and neither Z nor any of its descendants is in S

In words, S blocks a path when it blocks the flow of information by conditioning on the middle node in a chain or fork, and doesn't create dependence by conditioning on the middle node in a collider (or the descendant of a collider). Only *one* node in a path must be blocked to block the whole path. When S blocks *all* the paths between X and Y, we say it **d-separates** them[7]. A collection of variables U is d-separated from another collection V by S if every $X \in U$ and $Y \in V$ are d-separated.

In every distribution which obeys the Markov property, d-separation implies conditional independence[8]. It is not *always* the case that the reverse implication, the one from conditional independence to d-separation, holds good. We will see in Part III, that when the distribution is "faithful" to a DAG, causal inference is immensely simplified. But going from d-separation to conditional independence is true in any DAG, whether or not it has a causal interpretation.

18.3.1 D-Separation Illustrated

The discussion of d-separation has been rather abstract, and perhaps confusing for that reason. Figure 18.4 shows a DAG which might make this clearer and more concrete.

If we make the conditioning set S the empty set, that is, we condition on nothing, we "block" paths which pass through colliders. For instance, there are three exogenous variables in the graph, X_2, X_3 and X_5. Because they have no parents, any path from one to another must go over a collider (Exercises 18.1 and 18.2). If we do not condition on anything, therefore, we find that the exogenous variables are d-separated and thus independent. Since X_3 is not on any path linking X_2 and X_5, or descended from a node on any such path, if we condition only on X_3, then X_2 and X_5 are still d-separated, so $X_2 \perp\!\!\!\perp X_5 | X_3$. There are two paths linking X_3 to X_5: $X_3 \rightarrow X_1 \leftarrow X_2 \rightarrow X_4 \leftarrow X_5$, and $X_3 \rightarrow X_1 \rightarrow Y \leftarrow X_5$. Conditioning on X_2 (and nothing else) blocks the first path (since X_2 is part of it, but is a fork), and also blocks the second path (since X_2 is not part of it, and Y is a blocked collider). Thus, $X_3 \perp\!\!\!\perp X_5 | X_2$. Similarly, $X_3 \perp\!\!\!\perp X_2 | X_5$ (Exercise 18.4).

For a somewhat more challenging example, let's look at the relation between X_3 and Y. There are, again, two paths here: $X_3 \rightarrow X_1 \rightarrow Y$, and $X_3 \rightarrow X_1 \leftarrow X_2 \rightarrow X_4 \leftarrow X_5 \rightarrow Y$. If we condition on nothing, the first path, which is a simple chain, is open, so X_3 and Y are d-connected and dependent. If we condition on X_1, we block the first path. X_1 is a collider on the second path, so conditioning on X_1 opens the path there. However, there is a second collider, X_4, along this path, and just conditioning on X_1 does not activate the second collider, so the

[7] The "d" stands for "directed"

[8] We will not prove this, though I hope I have made it plausible. You can find demonstrations in Spirtes *et al.* (2001); Pearl (2000); Lauritzen (1996).

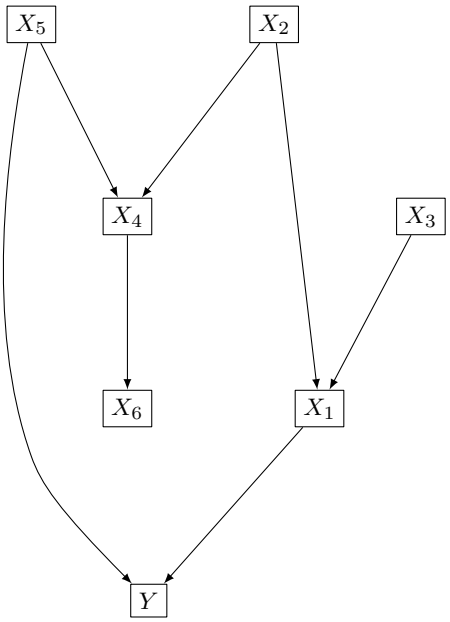


Figure 18.4 Example DAG used to illustrate d-separation.

path as a whole remains blocked.

$$Y \not\perp\!\!\!\perp X_3 \tag{18.8}$$
$$Y \perp\!\!\!\perp X_3 | X_1 \tag{18.9}$$

To activate the second path, we can condition on X_1 and either X_4 (a collider along that path) or on X_6 (a descendant of a collider) or on both:

$$Y \not\perp\!\!\!\perp X_3 | X_1, X_4 \tag{18.10}$$
$$Y \not\perp\!\!\!\perp X_3 | X_1, X_6 \tag{18.11}$$
$$Y \not\perp\!\!\!\perp X_3 | X_1, X_4, X_6 \tag{18.12}$$

Conditioning on X_4 and/or X_6 does not activate the $X_3 \rightarrow X_1 \rightarrow Y$ path, but it's enough for there to be one active path to create dependence.

To block the second path again, after having opened it in one of these ways, we can condition on X_2 (since it is a fork along that path, and conditioning on a fork blocks it), or on X_5 (also a fork), or on both X_2 and X_5. So

$$Y \perp\!\!\!\perp X_3 | X_1, X_2 \tag{18.13}$$
$$Y \perp\!\!\!\perp X_3 | X_1, X_5 \tag{18.14}$$
$$Y \perp\!\!\!\perp X_3 | X_1, X_2, X_5 \tag{18.15}$$
$$Y \perp\!\!\!\perp X_3 | X_1, X_2, X_4 \tag{18.16}$$
$$Y \perp\!\!\!\perp X_3 | X_1, X_2, X_6 \tag{18.17}$$
$$Y \perp\!\!\!\perp X_3 | X_1, X_2, X_5, X_6 \tag{18.18}$$

etc., etc.

Let's look at the relationship between X_4 and Y. X_4 is not an ancestor of Y, or a descendant of it, but they do share common ancestors, X_5 and X_2. Unconditionally, Y and X_4 are dependent, both through the path going $X_4 \leftarrow X_5 \rightarrow Y$, and through that going $X_4 \leftarrow X_2 \rightarrow X_1 \rightarrow Y$. Along both paths, the exogenous variables are forks, so *not* conditioning on them leaves the path unblocked. X_4 and Y become d-separated when we condition on X_5 and X_2.

X_6 and X_3 have no common ancestors. Unconditionally, they should be independent, and indeed they are: the two paths are $X_6 \leftarrow X_4 \leftarrow X_2 \rightarrow X_1 \leftarrow X_3$, and $X_6 \leftarrow X_4 \leftarrow X_5 \rightarrow Y \leftarrow X_1 \leftarrow X_3$. Both paths contain a single collider (X_1 and Y, respectively), so if we do not condition on them the paths are blocked and X_6 and X_3 are independent. If we condition on either Y or X_1 (or both), however, we unblock the paths, and X_6 and X_3 become d-connected, hence dependent. To get back to d-separation while conditioning on Y, we must also condition on X_4 or X_5, or both. To get d-separation while conditioning on X_1, we must also condition on X_4, or on X_2, or on X_4 and X_2. If we condition on both X_1 and Y and want d-separation, we could just add conditioning on X_4, or we could condition on X_2 and X_5, or all three.

If this is all still too abstract, consider reading the variables as follows:

$$
\begin{aligned}
Y &\Leftrightarrow \text{Grade in this class} \\
X_1 &\Leftrightarrow \text{Effort spent on this class} \\
X_2 &\Leftrightarrow \text{Enjoyment of statistics} \\
X_3 &\Leftrightarrow \text{Workload this term} \\
X_4 &\Leftrightarrow \text{Quality of work in linear regression class} \\
X_5 &\Leftrightarrow \text{Amount learned in linear regression class} \\
X_6 &\Leftrightarrow \text{Grade in linear regression}
\end{aligned}
$$

Pretending, for the sake of illustration, that this is accurate, how heavy your workload is this term (X_3) would predict, or rather retrodict, your grade in linear regression last term (X_6), once we control for how much effort you put into this class (X_1). Changing your workload this term would not, however, reach backwards in time to raise or lower your grade in regression.

18.3.2 Linear Graphical Models and Path Coefficients

We began our discussion of graphical models with factor analysis as our starting point. Factor models are a special case of linear (directed) graphical models, a.k.a. path models[9] As with factor models, in the larger class we typically center all the variables (so they have expectation zero) and scale them (so they have variance 1). In factor models, the variables were split into two sets, the factors and the observables, and all the arrows went from factors to observables. In the more

[9] Some people use the phrase "structural equation models" for linear directed graphical models exclusively.

general case, we do not necessarily have this distinction, but we still assume the arrows from a directed acyclic graph. The conditional expectation of each variable is a linear combination of the values of its parents:

$$\mathbb{E}\left[X_i \mid X_{\text{parents}(i)}\right] = \sum_{j \in \text{parents}(i)} w_{ji} X_j \tag{18.19}$$

just as in a factor model. In a factor model, the coefficients w_{ji} were the factor loadings. More generally, they are called **path coefficients**.

The path coefficients determine all of the correlations between variables in the model. If all of the variables have been standardized to mean zero and variance 1, and the path coefficients are calculated for these standardized variables, we can find the correlation between X_i and X_j as follows:

- Find all of the undirected paths between X_i and X_j.
- Discard all of the paths which go through colliders.
- For each remaining path, multiply all the path coefficients along the path.
- Sum up these products over paths.

These rules were introduced by the great geneticist and mathematical biologist Sewall Wright in the early 20th century (see further reading for details). These "Wright path rules" often seem mysterious, particularly the bit where paths with colliders are thrown out. But from our perspective, we can see that what Wright is doing is finding all of the *unblocked* paths between X_i and X_j. Each path is a channel along which information (here, correlation) can flow, and so we add across channels.

It is frequent, and customary, to assume that all of the variables are Gaussian. (We saw this in factor models as well.) With this extra assumption, the joint distribution of all the variables is a multivariate Gaussian, and the correlation matrix (which we find from the path coefficients) gives us the joint distribution.

If we want to find correlations conditional on a set of variables S, $\text{corr}(X_i, X_j|S)$, we still sum up over the unblocked paths. If we have avoided conditioning on colliders, then this is just a matter of dropping the now-blocked paths from the sum. If on the other hand we have conditioned on a collider, that path *does* become active (unless blocked elsewhere), and we in fact need to modify the path weights. Specifically, we need to work out the correlation induced between the two parents of the collider, by conditioning on that collider. This can be calculated from the path weights, and some fairly tedious algebra[10]. The important thing is to remember that the rule of d-separation still applies, and that conditioning on a collider can create correlations.

18.3.2.1 Path Coefficients and Covariances

If the variables have not all been standardized, but Eq. 18.19 still applies, it is often desirable to calculate covariances, rather than correlation coefficients. This involves a little bit of extra work, by way of keeping track of variances, and in

[10] See for instance Li *et al.* (1975).

particular the variances of "source" terms. Since many references do not state the path-tracing rules for covariances, it's worth going over them here.

To find the *marginal* covariance between X_i and X_j, the procedure is as follows:

1. Find all of the unblocked paths between X_i and X_j (i.e., discard all paths which go through colliders).
2. For each remaining path:
    1. multiply all the path coefficients along the path;
    2. find the node along that path which is the ancestor of all the other nodes along that path[11], and call it the path's source;
    3. multiply the product of the coefficients by the variance of the source.
3. Sum the product of path coefficients and source variances over all remaining paths.

(Notice that if all variables are standardized to variance 1, we don't have to worry about source variances, and these rules reduce to the previous ones.)

To find the *conditional* covariance between X_i and X_j given a set of variables S, there are two procedures, depending on whether or not conditioning on S opens any paths between X_i and X_j by including colliders. If S does not contain any colliders or descendants of colliders (on paths between X_i and X_j),

1. For each unblocked path linking X_i and X_j:
    1. multiply all the path coefficients along the path;
    2. find the source of each path[12];
    3. multiply the product of the coefficients by the variance of the source.
2. Sum the product of path coefficients and source variances over all remaining paths.

If, on the other hand, conditioning on S opens paths by conditioning on colliders (or their descendants), then we would have to handle the consequences of conditioning on a collider. This is usually too much of a pain to do graphically, and one should fall back on algebra. The next sub-section does however say a bit about what *qualitatively* happens to the correlations.

[[TODO: In final revision, write out full graphical rules for completeness]]

18.3.3 Positive and Negative Associations

We say that variables X and Y are **positively associated** if increasing X predicts, on average, an increase in Y, and vice versa[13]; if increasing X predicts a decrease in Y, then they are **negatively associated**. If this holds when conditioning out other variables, we talk about positive and negative partial associations. Heuristically, positive association means positive correlation in the neighborhood of any given x, though the magnitude of the positive correlation need not be

[11] Showing that such an ancestor exists is Exercise 21.

[12] Showing that the source of an unblocked, collider-free path cannot be in S is Exercise 22.

[13] I.e., if $\frac{d\mathbb{E}[Y|X=x]}{dx} \geq 0$

constant. Note that not all dependent variables have to have a definite sign for their association.

We can multiply together the signs of positive and negative partial associations along a path in a graphical model, the same we can multiply together path coefficients in a linear graphical model. Paths which contain (inactive!) colliders should be neglected. If all the paths connecting X and Y have the same sign, then we know that over-all association between X and Y must have that sign. If different paths have different signs, however, then signs alone are not enough to tell us about the over-all association.

If we are interested in conditional associations, we have to consider whether our conditioning variables block paths or not. Paths which are blocked by conditioning should be dropped from consideration. If a path contains an activated collider, we need to include it, but we reverse the sign of one arrow into the collider. That is, if $X \stackrel{+}{\rightarrow} Z \stackrel{+}{\leftarrow} Y$, and we condition on Z, we need to replace one of the plus signs with a $-$ sign, because the two parents now have an over-all negative association.[14] If on the other hand one of the incoming arrows had a positive association and the other was negative, we need to flip one of them so they are both positive or both negative; it doesn't matter which, since it creates a positive association between the parents[15].

[[TODO: Write out formal proofs as appendix]]

18.4 Independence, Conditional Independence, and Information Theory

Take two random variables, X and Y. They have some joint distribution, which we can write $p(x, y)$. (If they are both discrete, this is the joint probability mass function; if they are both continuous, this is the joint probability density function; if one is discrete and the other is continuous, there's still a distribution, but it needs more advanced tools.) X and Y each have marginal distributions as well, $p(x)$ and $p(y)$. $X \perp\!\!\!\perp Y$ if and only if the joint distribution is the product of the marginals:

$$X \perp\!\!\!\perp Y \Leftrightarrow p(x, y) = p(x)p(y) \tag{18.20}$$

We can use this observation to measure how dependent X and Y are. Let's start with the log-likelihood ratio between the joint distribution and the product of marginals:

$$\log \frac{p(x, y)}{p(x)p(y)} \tag{18.21}$$

[14] If both smoking and asbestos are positively associated with lung cancer, and we know the patient does not have lung cancer, then high levels of smoking must be compensated for by low levels of asbestos, and vice versa.

[15] If yellow teeth are positively associated with smoking and negatively associated with dental insurance, and we know the patient does not have yellow teeth, then high levels of smoking must be compensated for by excellent dental care, and conversely poor dental care must be compensated for by low levels of smoking.

This will always be exactly 0 when $X \perp\!\!\!\perp Y$. We use its average value as our measure of dependence:

$$I[X;Y] \equiv \sum_{x,y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{18.22}$$

(If the variables are continuous, replace the sum with an integral.) Clearly, if $X \perp\!\!\!\perp Y$, then $I[X;Y] = 0$. One can show[16] that $I[X;Y] \geq 0$, and that $I[X;Y] = 0$ implies $X \perp\!\!\!\perp Y$. The quantity $I[X;Y]$ is clearly symmetric between X and Y. Less obviously, $I[X;Y] = I[f(X);g(Y)]$ whenever f and g are invertible functions. This **coordinate-freedom** means that $I[X;Y]$ measures *all forms* of dependence, not just linear relationships, like the ordinary (Pearson) correlation coefficient, or monotone dependence, like the rank (Spearman) correlation coefficient. In information theory, $I[X;Y]$ is called the **mutual information**, or **Shannon information**, between X and Y. So we have the very natural statement that random variables are independent just when they have no information about each other.

There are (at least) two ways of giving an operational meaning to $I[X;Y]$. One, the original use of the notion, has to do with using knowledge of Y to improve the efficiency with which X can be encoded into bits (Shannon, 1948; Cover and Thomas, 2006). While this is very important — it's literally transformed the world since 1945 — it's not very statistical. For statisticians, what matters is that if we test the hypothesis that X and Y are independent, with joint distribution $p(x)p(y)$, against the hypothesis that they dependent, with joint distribution $p(x,y)$, then the mutual information controls the error probabilities of the test. To be exact, if we fix any power we like (90%, 95%, 99.9%, ...), the size or type I error rate α_n, of the best possible test shrinks exponentially with the number of IID samples n, and the rate of exponential decay is precisely $I[X;Y]$ (Kullback, 1968, §4.3, theorem 4.3.2):

$$\lim_{n\to\infty} -\frac{1}{n} \log \alpha_n \leq I[X;Y] \tag{18.23}$$

So positive mutual information means dependence, and the magnitude of mutual information tells us about how detectable the dependence is[17].

Suppose we conditioned X and Y on a third variable (or variables) Z. For each realization z, we can calculate the mutual information,

$$I[X;Y|Z=z] \equiv \sum_{x,y} p(x,y|z) \log \frac{p(x,y|z)}{p(x|z)p(y|z)} \tag{18.24}$$

[16] Using the same type of convexity argument ("Jensen's inequality") we used §17.2.1 for understanding why the EM algorithm works.

[17] Symmetrically, if we follow the somewhat more usual procedure of fixing a type I error rate α, the type II error rate β_n (= 1−power) also goes to zero exponentially, and the exponential rate is $\sum_{x,y} p(x)p(y) \log \frac{p(x)p(y)}{p(x,y)}$, a quantity called the "lautam information" (Palomar and Verdú, 2008). (For proofs of the exponential rate, see Palomar and Verdú (2008, p. 965), following Kullback (1968, §4.3, theorem 4.3.3).)



Figure 18.5 DAG for a mixture model. The latent class Z is exogenous, and the parent of the observable random vector X. (If the components of X are conditionally independent given Z, they could be represented as separate boxes on the lower level.

And we can average over z,

$$I[X;Y|Z] \equiv \sum_z p(z) I[X;Y|Z=z] \tag{18.25}$$

This is the **conditional mutual information**. It will not surprise you at this point to learn that $X \perp\!\!\!\perp Y|Z$ if and only if $I[X;Y|Z] = 0$. The magnitude of the conditional mutual information tells us how easy it is to detect conditional dependence.

18.5 Examples of DAG Models and Their Uses

Factor models are examples of DAG models (as we've seen). So are mixture models (Figure 18.5) and Markov chains (see above). DAG models are considerably more flexible, however, and can combine observed and unobserved variables in many ways.

Consider, for instance, Figure 18.6. Here there are two exogeneous variables, labeled "Smoking" and "Asbestos". Everything else is endogenous. Notice that "Yellow teeth" is a child of "Smoking" alone. This does not mean that (in the model) whether someone's teeth get yellowed (and, if so, how much) is a function of smoking alone; it means that whatever other influences go into that are independent of the rest of the model, and so unsystematic that we can think about those influences, taken together, as noise.

Continuing, the idea is that how much someone smokes influences how yellow their teeth become, and also how much tar builds up in their lungs. Tar in the lungs, in turn, leads to cancer, as does by exposure to asbestos.

Now notice that, in this model, teeth-yellowing will be unconditionally dependent on, i.e., associated with, the level of tar in the lungs, because they share a common parent, namely smoking. Yellow teeth and tarry lungs will however be conditionally independent given that parent, so if we control for smoking we should not be able to predict the state of someone's teeth from the state of their lungs or vice versa.

On the other hand, smoking and exposure to asbestos are independent, at least

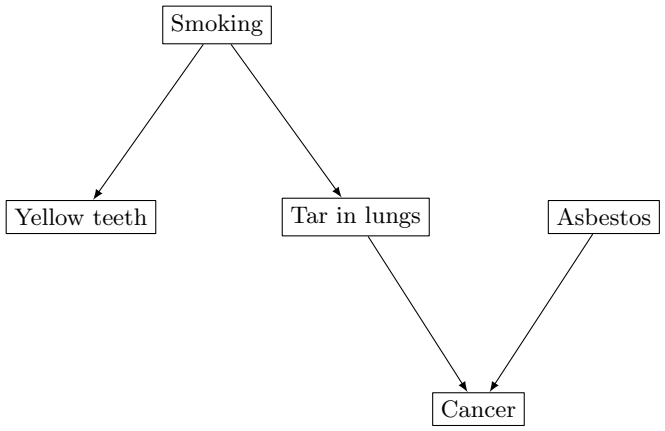


Figure 18.6 DAG model indicating (hypothetical) relationships between smoking, asbestos, cancer, and covariates.

in this model, as they are both exogenous[18]. Conditional on whether someone has cancer, however, smoking and asbestos will become *dependent*.

To understand the logic of this, suppose (what is in fact true) that both how much someone smokes and how much they are exposed to asbestos raises the risk of cancer. Conditional on not having cancer, then, one was probably exposed to little of either tobacco smoke or asbestos. Conditional on both not having cancer and having been exposed to a high level of asbestos, one probably was exposed to an unusually low level of tobacco smoke. Vice versa, no cancer plus high levels of tobacco tend to imply especially little exposure to asbestos. We thus have created a negative association between smoking and asbestos by conditioning on cancer. Naively, a regression where we "controlled for" cancer would in fact tell us that exposure to asbestos keeps tar from building up in the lungs, prevents smoking, and whitens teeth.

More generally, conditioning on a third variable can *create* dependence between otherwise independent variables, when what we are conditioning on is a common *descendant* of the variables in question.[19] This conditional dependence is *not* some kind of finite-sample artifact or error — it's really there in the joint probability distribution. If all we care about is prediction, then it is perfectly legitimate to use it. In the world of Figure 18.6, it really is true that you can predict the color of someone's teeth from whether they have cancer and how much

[18] If we had two variables which in some physical sense were exogenous but dependent on each other, we would represent them in a DAG model by either a single *vector-valued* random variable (which would get only one node), or as children of a latent unobserved variable, which was truly exogenous.

[19] Economists, psychologists, and other non-statisticians often repeat the advice that if you want to know the effect of X on Y, you should not condition on Z when Z is endogenous. This is bit of folklore is a relic of the days of ignorance, when our ancestors groped towards truths they could not grasp. If we want to know whether asbestos is associated with tar in the lungs, conditioning on the yellowness of teeth is fine, even though that is an endogenous variable.

asbestos they've been exposed to, so if that's what you want to predict[20], why not use that information? But if you want to do more than just make predictions without understanding, if you want to understand the structure tying together these variables, if you want to do *science*, if you don't want to go around telling yourself that asbestos whitens teeth, you really do need to know the graph.[21]

18.5.1 Missing Variables

Suppose that we do not observe one of the variables, such as the quantity of tar in the lungs, but we somehow know all of the conditional distributions required by the graph. (Tar build-up in the lungs might indeed be hard to measure for living people.) Because we have a joint distribution for *all* the variables, we could estimate the conditional distribution of one of them given the rest, using the definition of conditional probability and of integration:

$$p(X_i|X_1, X_2, X_{i-1}, X_{i+1}, X_p) = \frac{p(X_1, X_2, X_{i-1}, X_i, X_{i+1}, X_p)}{\int p(X_1, X_2, X_{i-1}, x_i, X_{i+1}, X_p) dx_i} \quad (18.26)$$

We could in principle do this for *any* joint distribution. When the joint distribution comes from a DAG model, however, we can simplify this considerably. Recall from §18.2.1 that X_i is independent of all the other variables given its Markov blanket, i.e., its parents, its children, and the other parents of its children. We can therefore drop from the conditioning everything which isn't in the Markov blanket. Actually *doing* the calculation then boils down to a version of the EM algorithm.[22]

If we observe only a subset of the other variables, we can still use the DAG to determine which ones actually matter to estimating X_i, and which ones are superfluous. The calculations then however become much more intricate.[23]

18.6 Non-DAG Graphical Models: Undirected Graphs and Directed Graphs with Cycles

For various reasons (many of them explained below!), we will not use these models in the rest of this book.

18.6.1 Undirected Graphs

There is a lot of work on probability models which are based on *undirected* graphs, in which the relationship between random variables linked by edges is completely

[20] Maybe you want to guess who'd be interested in buying whitening toothpaste.

[21] We return to this example in §19.2.2.

[22] Graphical models, especially directed ones, are often called "Bayes nets" or "Bayesian networks", because this equation is, or can be seen as, a version of Bayes's rule. Since of course it follows directly from the definition of conditional probability, there is nothing distinctively Bayesian here — no subjective probability, or assigning probabilities to hypotheses.

[23] There is an extensive discussion of relevant methods in Jordan (1998).

symmetric, unlike the case of DAGs[24]. Since the relationship is symmetric, the preferred metaphor is not "parent and child", but "neighbors". The models are sometimes called **Markov networks** or **Markov random fields**, but since DAG models have a Markov property of their own, this is not a happy choice of name, and I'll just call them "undirected graphical models".

The key Markov property for undirected graphical models is that any set of nodes I is independent of the rest of the graph given its neighbors:

$$X_I \perp\!\!\!\perp X_{\text{non-neighbors}(I)} | X_{\text{neighbors}(I)} \tag{18.27}$$

This corresponds to a factorization of the joint distribution, but a more complex one than that of Eq. 18.4, because a symmetric neighbor-of relation gives us no way of *ordering* the variables, and conditioning the later ones on the earlier ones. The trick turns out to go as follows. First, as a bit of graph theory, a **clique** is a set of nodes which are all neighbors of each other, and which cannot be expanded without losing that property. We write the collection of all cliques in a graph G as cliques(G). Second, we introduce functions ψ_c which take clique configurations and return non-negative numbers. Third, we say that a joint distribution is a **Gibbs distribution**[25] when

$$p(X_1, X_2, \ldots X_p) \propto \prod_{c \in \text{cliques}(G)} \psi_c(X_{i \in c}) \tag{18.28}$$

That is, the joint distribution is a product of factors, one factor for each clique. Frequently, one introduces what are called **potential functions**, $U_c = \log \psi_c$, and then one has

$$p(X_1, X_2, \ldots X_p) \propto e^{-\sum_{c \in \text{cliques}(G)} U_i(X_{i \in c})} \tag{18.29}$$

The key correspondence is what may be called the **Gibbs-Markov theorem**: a distribution is a Gibbs distribution with respect to a graph G if, and only if, it obeys the Markov property with neighbors defined according to G.[26].

In many practical situations, one combines the assumption of an undirected graphical model with the further assumption that the joint distribution of all the random variables is a multivariate Gaussian, giving a **Gaussian graphical**

[24] I am told that this is more like the idea of causation in Buddhism, as something like "co-dependent origination", than the asymmetric one which Europe and the Islamic world inherited from the Greeks (especially Aristotle), but you would really have to ask a philosopher about that.

[25] After the American physicist and chemist J. W. Gibbs, who introduced such distributions as part of **statistical mechanics**, the theory of the large-scale patterns produced by huge numbers of small-scale interactions.

[26] This theorem was proved, in slightly different versions, under slightly different conditions, and by very different methods, more or less simultaneously by (alphabetically) Dobrushin, Griffeath, Grimmett, and Hammersley and Clifford, and almost proven by Ruelle. In the statistics literature, it has come to be called the "Hammersley-Clifford" theorem, for no particularly good reason. In my opinion, the clearest and most interesting version of the theorem is that of Griffeath (1976), an elementary exposition of which is given by Pollard (`http://www.stat.yale.edu/~pollard/Courses/251.spring04/Handouts/Hammersley-Clifford.pdf`). (On the other hand, Griffeath was one of my teachers, so discount accordingly.) Calling it the "Gibbs-Markov theorem" says more about the content, and is fairer to all concerned.

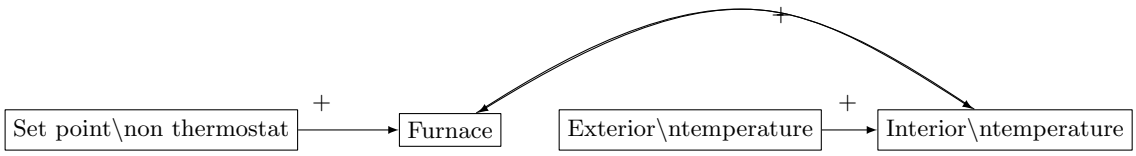


Figure 18.7 Directed but cyclic graphical model of a feedback loop. Signs (+, − on arrows are "guides to the mind". Cf. Figure 18.8.

model. An important consequence of this assumption is that the graph can be "read off" from the inverse of the covariance matrix $\mathbf{\Sigma}$, sometimes called the **precision matrix**. Specifically, there is an edge linking X_i to X_j if and only if $(\mathbf{\Sigma}^{-1})_{ij} \neq 0$. (See Lauritzen (1996) for an extensive discussion.) These ideas sometimes still work for non-Gaussian distributions, when there is a natural way of transforming them to be Gaussian (Liu *et al.*, 2009), though it is unclear just how far that goes.

18.6.2 Directed but Cyclic Graphs

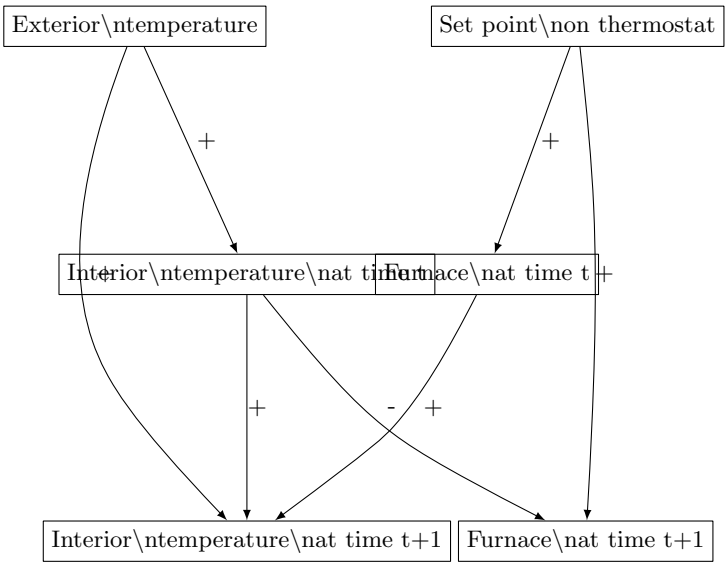
Much less work has been done on directed graphs with cycles. It is very hard to give these a causal interpretation, in the fashion described in the next chapter. Feedback processes are of course very common in nature and technology, and one might think to represent these as cycles in a graph. A model of a thermostat, for instance, might have variables for the set-point temperature, the temperature outside, how much the furnace runs, and the actual temperature inside, with a cycle between the latter two (Figure 18.7).

Thinking in this way is however simply sloppy. It always takes *some* time to traverse a feedback loop, and so the cycle really "unrolls" into an acyclic graph linking similar variables at *different* times (Figure 18.8). Sometimes[27], it is clear that when people draw a diagram like Figure 18.7, the incoming arrows really refer to the change, or rate of change, of the variable in question, so it is merely a visual short-hand for something like Figure 18.8.

Directed graphs with cycles are thus primarily useful when measurements are so slow or otherwise imprecise that feedback loops cannot be unrolled into the actual dynamical processes which implement them, and one is forced to hope that one can reason about equilibria instead[28]. If you insist on dealing with cyclic directed graphical models, see Richardson (1996); Lacerda *et al.* (2008) and references therein.

[27] As in Puccia and Levins (1985), and the `LoopAnalyst` package based on it (Dinno, 2009).

[28] Economists are fond of doing so, generally without providing any rationale, based in economic theory, for supposing that equilibrium *is* a good approximation (Fisher, 1983, 2010).



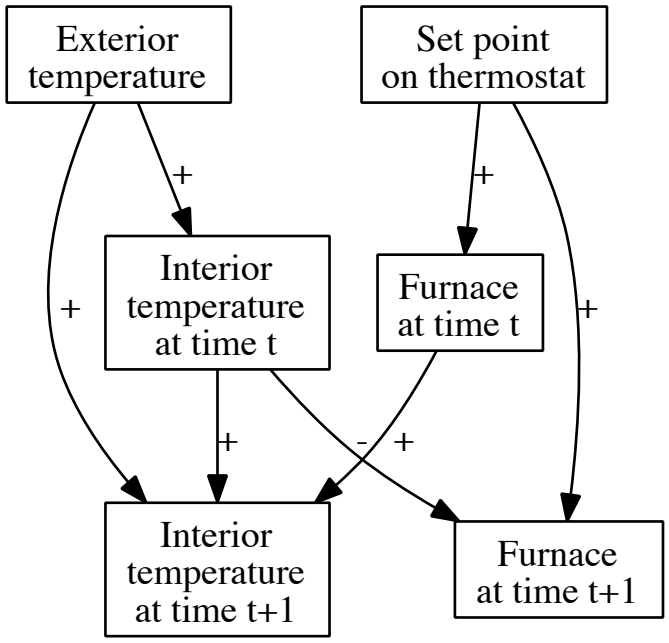


Figure 18.8 Directed, acyclic graph for the situation in Figure 18.7, taking into account the fact that it takes time to traverse a feedback loop. One should imagine this repeating to times $t + 2$, $t + 3$, etc., and extending backwards to times $t - 1$, $t - 2$, etc., as well. Notice that there are no longer any cycles.

18.7 Further Reading

The paper collection Jordan (1998) is actually extremely good, unlike most collections of edited papers; Jordan and Sejnowski (2001) is also useful. Lauritzen (1996) is thorough but more mathematically demanding. The books by Spirtes *et al.* (1993, 2001) and by Pearl (1988, 2000, 2009b) are deservedly classics, especially for their treatment of causality, of which much more in Part III. Glymour (2001) discusses applications to psychology.

While I have presented DAG models as an outgrowth of factor analysis, their historical ancestry is actually closer to the "path analysis" models introduced, starting around 1918, by the great geneticist and mathematical biologist Sewall Wright to analyze processes of development and genetics. Wright published his work in a series of papers which culminated in Wright (1934). That paper is now freely available online, and worth reading. (See also `http://www.ssc.wisc.edu/soc/class/soc952/Wright/wright_biblio.htm` for references to, and in some cases copies of, related papers by Wright.) Path analysis proved extremely influential in psychology and sociology. Loehlin (1992) is user-friendly, though aimed at psychologists who know less math anyone taking this course. Li (1975), while older, is very enthusiastic and has many interesting applications in biology. Moran (1961) is a very clear treatment of the mathematical foundations, extended by Wysocki (1992) to the case where each variable is itself multi-dimensional vector, so that path "coefficients" are themselves matrices.

Markov random fields where the graph is a regular lattice are used extensively in spatial statistics. Good introductory-level treatments are provided by Kindermann and Snell (1980) (the full text of which is free online), and by Guttorp (1995), which also covers the associated statistical methods. Winkler (1995) is also good, but presumes more background in statistical theory. (I would recommend reading it after Guttorp.) Griffeath (1976), while presuming more probability theory on the part of the reader, is extremely clear and insightful, including what is simultaneously one of the deepest and most transparent proofs of the Gibbs-Markov theorem. Lauritzen (1996) is a mathematically rigorous treatment of graphical models from the viewpoint of theoretical statistics, covering both the directed and undirected cases.

If you are curious about Gibbs distributions in their (so to speak) natural habitat, the book by Sethna (2006), also free online, is the best introduction to statistical mechanics I have seen, and presumes very little knowledge of actual physics on the part of the reader. Honerkamp (2002) is less friendly, but tries harder to make connections to statistics. If you already know what an exponential family is, then Eq. 18.29 is probably extremely suggestive, and you should read Mandelbrot (1962).

On information theory (§18.4), the best book is Cover and Thomas (2006) by a large margin. References specifically on the connection between *causal* graphical models and information theory are given in Chapter 19.

Exercises

18.1 Find all the paths between the exogenous variables in Figure 18.4, and verify that every such path goes through at least one collider .

18.2 Is it true that in any DAG, every path between exogenous variables must go through at least one collider, or descendant of a collider? Either prove it or construct a counter-example in which it is not true. Does the answer change we say "go through at least one collider", rather than "collider or descendant of a collider"? .

18.3
1. Take any two nodes, say X_1 and X_2, which are linked in a DAG by a path which does not go over colliders. Prove that there is a unique node along the path which is an ancestor of all other nodes on that path. (Note that this shared ancestor may in fact be X_1 or X_2.) *Hint:* do exercise 18.2.
2. Take any two nodes which are linked in a DAG by a path which remains open when conditioning on a set of variables S containing no colliders. Prove that for every open path between X_1 and X_2, there is a unique node along the path which is an ancestor of all other nodes on that path, and that this ancestor is not in S.

18.4 Prove that $X_2 \perp\!\!\!\perp X_3 | X_5$ in Figure 18.4.

Part III

Causal Inference

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

19

Graphical Causal Models

19.1 Causation and Counterfactuals

Take a piece of cotton, say an old rag. Apply flame to it; the cotton burns. We say the fire *caused* the cotton to burn. The flame is certainly *correlated* with the cotton burning, but, as we all know, correlation is not causation (Figure 19.1). Perhaps every time we set rags on fire we handle them with heavy protective gloves; the gloves don't make the cotton burn, but the statistical dependence is strong. So what is causation?

We do not have to settle 2500 years (or more) of argument among philosophers and scientists. For our purposes, it's enough to realize that the concept has a **counter-factual** component: if, contrary to fact, the flame had not been applied to the rag, then the rag would not have burned[1]. On the other hand, the fire makes the cotton burn whether we are wearing protective gloves or not.

To say it a somewhat different way, the distributions we observe in the world

[1] If you immediately start thinking about quibbles, like "What if we hadn't applied the flame, but the rag was struck by lightning?", then you may have what it takes to be a philosopher.

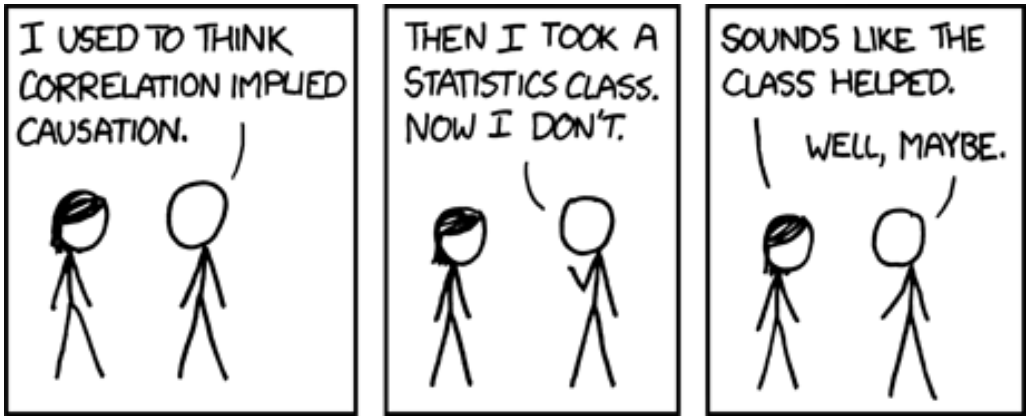

I USED TO THINK CORRELATION IMPLIED CAUSATION.
THEN I TOOK A STATISTICS CLASS. NOW I DON'T.
SOUNDS LIKE THE CLASS HELPED.
WELL, MAYBE.

Figure 19.1 "Correlation doesn't imply causation, but it does waggle its eyebrows suggestively and gesture furtively while mouthing 'look over there"' (Image and text copyright by Randall Munroe, used here under a Creative Commons attribution-noncommercial license; see `http://xkcd.com/552/`. [[TODO: Excise from the commercial version]])

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

are the outcome of complicated stochastic processes. The mechanisms which set the value of one variable inter-lock with those which set other variables. When we make a probabilistic prediction by conditioning — whether we predict $\mathbb{E}[Y \mid X = x]$ or $\Pr(Y \mid X = x)$ or something more complicated — we are just filtering the output of those mechanisms, picking out the cases where they happen to have set X to the value x, and looking at what goes along with that.

When we make a *causal* prediction, we want to know what would happen if the usual mechanisms controlling X were suspended and it was *set* to x. How would this change propagate to the other variables? What distribution would result for Y? This is often, perhaps even usually, what people really want to know from a data analysis, and they settle for statistical prediction either because they think it *is* causal prediction, or for lack of a better alternative.

Causal inference is the undertaking of trying to answer causal questions from empirical data. Its fundamental difficulty is that we are trying to derive counterfactual conclusions with only factual premises. As a matter of habit, we come to expect cotton to burn when we apply flames. We might even say, on the basis of purely statistical evidence, that the world has this habit. But as a matter of pure logic, no amount of evidence about what *did* happen can compel beliefs about what *would* have happened under non-existent circumstances[2]. (For all my *data* shows, all the rags I burn just so happened to be on the verge of spontaneously bursting into flames anyway.) We must supply some counter-factual or causal premise, linking what we see to what we could have seen, to derive causal conclusions.

One of our goals, then, in causal inference will be to make the causal premises as weak and general as possible, thus limiting what we take on faith.

19.2 Causal Graphical Models

We will need a formalism for representing causal relations. It will not surprise you by now to learn that these will be graphical models. We will in fact use DAG models from last time, with "parent" interpreted to mean "directly causes". These will be **causal graphical models**, or **graphical causal models**.[3]

We make the following assumptions.

1. There is some directed acyclic graph G representing the relations of causation among the our variables.

[2] The first person to really recognize this seems to have been the medieval Muslim theologian and anti-philosopher al Ghazali (1100/1997). (See Kogan (1985) for some of the history.) Very similar arguments were made centuries later by Hume (1739); whether there was some line of intellectual descent linking them — that is, any causal connection — I don't know.

[3] Because DAG models have joint distributions which factor according to the graph, we can always write them in the form of a set of equations, as $X_i = f_i(X_{\text{parents}(i)}) + \epsilon_i$, with the catch that the noise ϵ_i is not necessarily independent of X_i's parents. This is what is known, in many of the social sciences, as a **structural equation model**. So those are, strictly, a sub-class of DAG models. They are also often used to represent causal structure.

2. **The Causal Markov condition:** The joint distribution of the variables obeys the Markov property on G.
3. **Faithfulness:** The joint distribution has all of the conditional independence relations implied by the causal Markov property, and *only* those conditional independence relations.

The point of the faithfulness condition is to rule out "conspiracies among the parameters", where, say, two causes of a common effect, which would typically be dependent conditional on that effect, have their impact on the joint effect and their own distributions matched just so exactly that they remain conditionally independent.

19.2.1 Calculating the "effects of causes"

Let's fix two sub-sets of variables in the graph, X_c and X_e. (Assume they don't overlap, and call everything else X_N.) If we want to make a *probabilistic* prediction for X_e's value when X_c takes a particular value, x_c, that's the conditional distribution, $\Pr(X_e \mid X_c = x_c)$, and we saw last time how to calculate that using the graph. Conceptually, this amounts to selecting, out of the whole population or ensemble, the sub-population or sub-ensemble where $X_c = x_c$, and accepting whatever other behavior may go along with that.

Now suppose we want to ask what the effect would be, causally, of setting X_c to a particular value x_c. We represent this by "doing surgery on the graph": we (i) eliminate any arrows coming in to nodes in X_c, (ii) fix their values to x_c, and (iii) calculate the resulting distribution for X_e in the new graph. By steps (i) and (ii), we imagine suspending or switching off the mechanisms which ordinarily set X_c. The other mechanisms in the assemblage are left alone, however, and so step (iii) propagates the fixed values of X_c through them. We are not *selecting* a sub-population, but producing a new one.

If setting X_c to different values, say x_c and x_c', leads to different distributions for X_e, then we say that X_c **has an effect** on X_e — or, slightly redundantly, **has a causal effect** on X_e. Sometimes[4] "the effect of switching from x_c to x_c'" specifically refers to a change in the expected value of X_e, but since profoundly different distributions can have the same mean, this seems needlessly restrictive.[5] If one is interested in average effects of this sort, they are computed by the same procedure.

It is convenient to have a short-hand notation for this procedure of causal conditioning. One more-or-less standard idea, introduced by Judea Pearl, is to introduce a *do* operator which encloses the conditioning variable and its value. That is,

$$\Pr(X_e \mid X_c = x_c) \tag{19.1}$$

[4] Especially in economics.

[5] Economists are also fond of the horribly misleading usage of talking about "an X effect" or "the effect of X" when they mean the regression coefficient of X. Don't do this.

is probabilistic conditioning, or selecting a sub-ensemble from the old mechanisms; but

$$\Pr(X_e \mid do(X_c = x_c)) \tag{19.2}$$

is causal conditioning, or producing a new ensemble. Sometimes one sees this written as $\Pr(X_e \mid X_c \hat{=} x_c)$, or even $\Pr(X_e \mid \widehat{x_c})$. I am actually fond of the *do* notation and will use it.

Suppose that $\Pr(X_e \mid X_c = x_c) = \Pr(X_e \mid do(X_c = x_c))$. This would be extremely convenient for causal inference. The conditional distribution on the right is the causal, counter-factual distribution which tells us what would happen if x_c was imposed. The distribution on the left is the ordinary probabilistic distribution we have spent years learning how to estimate from data. When do they coincide?

One situation where they coincide is when X_c contains all the parents of X_e, and none of its descendants. Then, by the Markov property, X_e is independent of all other variables given X_c, and removing the arrows *into* X_c will not change that, or the conditional distribution of X_e given its parents. Doing causal inference for other choices of X_c will demand other conditional independence relations implied by the Markov property. This is the subject of Chapter 20.

19.2.2 Back to Teeth

Let us return to the example of Figure 18.6, and consider the relationship between exposure to asbestos and the staining of teeth. In the model depicted by that figure, the joint distribution factors as

$$\begin{aligned} & p(\text{Yellow teeth}, \text{Smoking}, \text{Asbestos}, \text{Tar in lungs}, \text{Cancer}) \\ & = p(\text{Smoking})p(\text{Asbestos}) \\ & \quad \times p(\text{Tar in lungs}|\text{Smoking}) \\ & \quad \times p(\text{Yellow teeth}|\text{Smoking}) \\ & \quad \times p(\text{Cancer}|\text{Asbestos}, \text{Tar in lungs}) \end{aligned} \tag{19.3}$$

As we saw, whether or not someone's teeth are yellow (in this model) is unconditionally independent of asbestos exposure, but conditionally *dependent* on asbestos, given whether or not they have cancer. A logistic regression of tooth color on asbestos would show a non-zero coefficient, after "controlling for" cancer. This coefficient would become significant with enough data. The usual interpretation of this coefficient would be to say that the log-odds of yellow teeth increase by so much for each one unit increase in exposure to asbestos, "other variables being held equal".[6] But to see the actual causal effect of increasing exposure to asbestos by one unit, we'd want to compare $p(\text{Yellow teeth}|do(\text{Asbestos} = a))$ to $p(\text{Yellow teeth}|do(\text{Asbestos} = a + 1))$, and it's easy to check (Exercise 19.1) that

[6] Nothing hinges on this being a logistic regression, similar interpretations are given to all the other standard models.

these two distributions have to be the same. In this case, because asbestos is exogenous, one will in fact get the same result for $p(\text{Yellow teeth}|do(\text{Asbestos} = a)$ and for $p(\text{Yellow teeth}|\text{Asbestos} = a)$.

For a more substantial example, consider Figure 19.2[7] The question of interest here is whether regular brushing and flossing actually prevents heart disease. The mechanism by which it might do so is as follows: brushing is known to make it less likely for people to get gum disease. Gum disease, in turn, means the gums suffer from constant, low-level inflammation. Persistent inflammation (which can be measured through various messenger chemicals of the immune system) is thought to increase the risk of heart disease. Against this, people who are generally health-conscious are likely to brush regularly, and to take other actions, like regularly exercising and controlling their diets, which also make them less likely to get heart disease. In this case, if we were to manipulate whether people brush their teeth[8], we would shift the graph from Figure 19.2 to Figure 19.3, and we would have

$$p(\text{Heart disease}|\text{Brushing} = b) \neq p(\text{Heart disease}|do(\text{Brushing} = b)) \quad (19.4)$$

19.3 Conditional Independence and d-Separation Revisited

We saw in §18.3 that all distributions which conform to a common DAG share a common set of conditional independence relations. Faithful distributions have *no other* conditional independence relations. These are vital facts for causal inference.

The reason is that while *causal influence* flows one way through the graph, along the directions of arrows from parents to children, *statistical information* can flow in either direction. We can certainly make inferences about an effect from its causes, but we can equally make inferences about causes from their effects. It might be harder to actually do the calculations[9], and we might be left with more uncertainty, but we could do it. As we saw in §18.3, when conditioning on a set of variables S blocks all channels of information flow between X and Y, $X \perp\!\!\!\perp Y|S$. The faithful distributions are the ones where this implication is reversed, where $X \perp\!\!\!\perp Y|S$ implies that S blocks all paths between X and Y. In faithful graphical models, blocking information flow is exactly the same as conditional independence.

This turns out to be the single most important fact enabling causal inference. If we want to estimate the effects of causes, within a given DAG, we need to block off all non-causal channels of information flow. If we want to check whether a given DAG is correct for the variables we have, we need to be able to compare

[7] Based on de Oliveira *et al.* (2010), and the discussion of this paper by Chris Blattman (`http://chrisblattman.com/2010/06/01/does-brushing-your-teeth-lower-cardiovascular-disease/`).

[8] Hopefully, by ensuring that everyone brushes, rather than keeping people from brushing.

[9] Janzing (2007) [[TODO: update refs]] makes the very interesting suggestion that the direction of causality can be discovered by using this — roughly speaking, that if $X|Y$ is much harder to compute than is $Y|X$, we should presume that $X \to Y$ rather than the other way around.

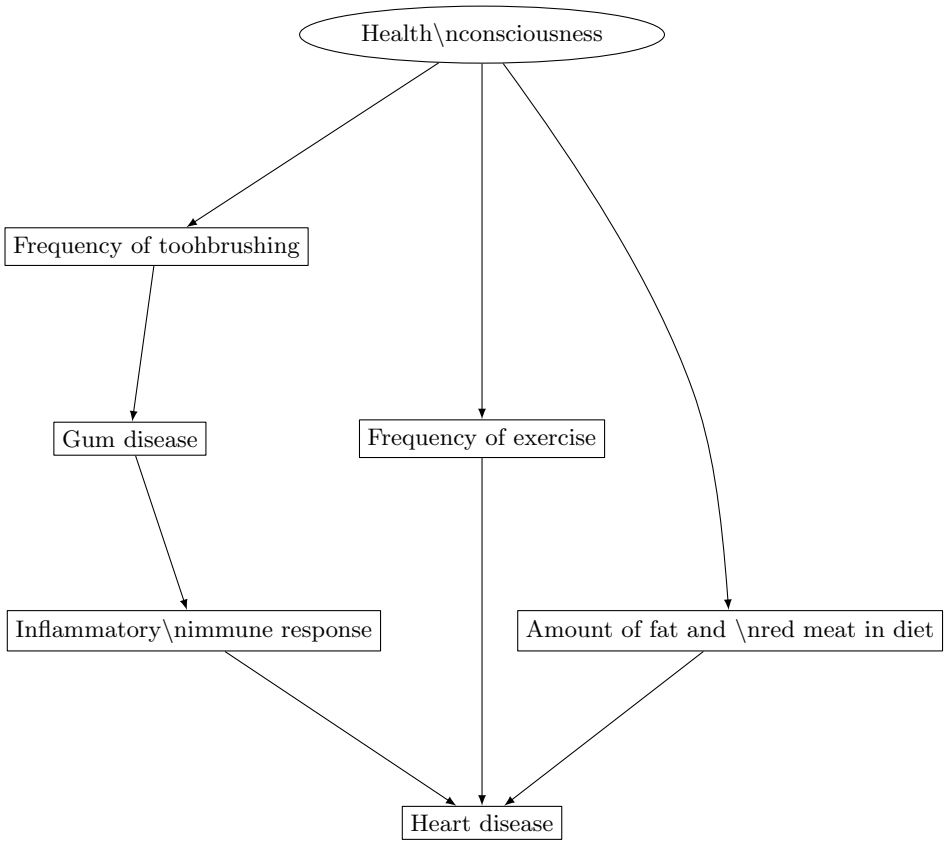


Figure 19.2 Graphical model illustrating hypothetical pathways linking brushing your teeth to not getting heart disease.

the conditional independence relations implied by the DAG to those supported by the data. If we want to discover the possible causal structures, we have to see which ones imply the conditional independencies supported by the data.

19.4 Further Reading

The two foundational books on graphical causal models are Spirtes *et al.* (2001) and Pearl (2009b). Both are excellent and recommended in the strongest possible terms; but if you had to read just one, I would recommend Spirtes *et al.* (2001). If on the other hand you do not feel up to reading a book at all, then Pearl (2009a) is much shorter, and covers the high points. (Also, it's free online.) The textbook by Morgan and Winship (2007, 2015) is much less demanding mathematically, and therefore also less complete conceptually, but it does explain the crucial ideas clearly, simply, and with abundant examples.[10] Lauritzen (1996) has

[10] That textbook also discusses an alternative formalism for counterfactuals, due mainly to Donald B. Rubin and collaborators. While Rubin has done very distinguished work in causal inference, his

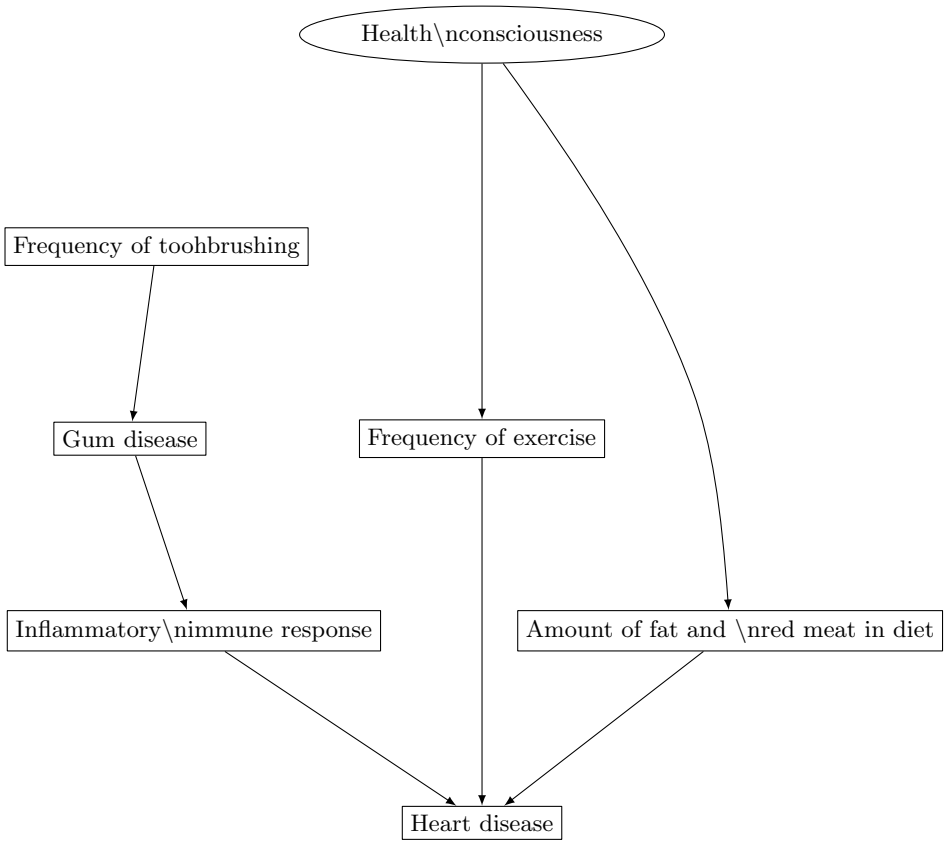


Figure 19.3 The previous graphical model, "surgically" altered to reflect a manipulation (*do*) of brushing.

a mathematically rigorous treatment of d-separation (among many other things), but de-emphasizes causality.

Many software packages for linear structural equation models and path analysis offer options to search for models; these are not, in general, reliable (Spirtes *et al.*, 2001).

Raginsky (2011) provides a fascinating information-theoretic account of graphical causal models and *do*(), in terms of the notion of directed (rather than mutual) information.

formalism is vastly harder to manipulate than are graphical models, but has no more expressive power. (Pearl (2009a) has a convincing discussion of this point, and Richardson and Robins (2013) provides a comprehensive proof that the everything expressible in the counterfactuals formalism can also be expressed with suitably-augmented graphical models.) I have thus skipped the Rubin formalism here, but there are good accounts in Morgan and Winship (2007, ch. 2), in Rubin's collected papers (Rubin, 2006), and in Imbens and Rubin (2015) (though please read Shalizi 2016 before taking any of the real-data examples in the last of these as models to imitate).

Exercises

19.1 Show, for the graphical model in Figure 18.6, that $p(\text{Yellow teeth}|do(\text{Asbestos} = a))$ is always the same as $p(\text{Yellow teeth}|do(\text{Asbestos} = a + 1))$.

20

Identifying Causal Effects from Observations

There are two problems which are both known as "causal inference":

1. Given the causal structure of a system, estimate the effects the variables have on each other.
2. Given data about a system, find its causal structure.

The first problem is easier, so we'll begin with it; we come back to the second in Chapter 22.

20.1 Causal Effects, Interventions and Experiments

As a reminder, when I talk about the causal effect of X on Y, which I write

$$\Pr(Y|do(X = x)) \tag{20.1}$$

I mean the distribution of Y which would be generated, counterfactually, were X to be set to the particular value x. This is not, in general, the same as the ordinary conditional distribution

$$\Pr(Y|X = x) \tag{20.2}$$

The reason these are different is that the latter represents taking the original population, as it is, and just filtering it to get the sub-population where $X = x$. The processes which set X to that value may also have influenced Y through other channels, and so this distribution will not, typically, really tell us what would happen if we reached in and manipulated X. We can sum up the contrast in a little table (Table 20.1). As we saw in Chapter 18, if we have the full graph for a directed acyclic graphical model, it tells us how to calculate the joint distribution of all the variables, from which of course the conditional distribution of any one variable given another follows. As we saw in Chapter 19, calculations of $\Pr(Y|do(X = x))$ use a "surgically" altered graph, in which all arrows into X are removed, and its value is pinned at x, but the rest of the graph is as before. If we know the DAG, and we know the distribution of each variable given its parents, we can calculate any causal effect we want, by graph-surgery.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

Probabilistic conditioning	Causal conditioning
$\Pr(Y\|X=x)$	$\Pr(Y\|do(X=x))$
Factual	Counter-factual
Select a sub-population	Generate a new population
Predicts passive observation	Predicts active manipulation
Calculate from full DAG	Calculate from surgically-altered DAG
Always identifiable when X and Y are observable	Not always identifiable even when X and Y are observable

Table 20.1 *Contrasts between ordinary probabilistic conditioning and causal conditioning. (See below on identifiability.)*

20.1.1 The Special Role of Experiment

If we want to estimate $\Pr(Y|do(X=x))$, the most reliable procedure is also the simplest: actually manipulate X to the value x, and see what happens to Y. (As my mother used to say, "Why think, when you can just do the experiment?") A causal or counter-factual assumption is still required here, which is that the *next* time we repeat the manipulation, the system will respond similarly, but this is pretty weak as such assumptions go.

While this seems like obvious common sense to us now, it is worth taking a moment to reflect on the fact that systematic experimentation is a very recent thing; it only goes back to around 1600. Since then, the knowledge we have acquired by combining experiments with mathematical theories have totally transformed human life, but for the first four or five thousand years of civilization, philosophers and sages much smarter than (almost?) any scientist now alive would have dismissed experiment as something fit only for cooks, potters and blacksmiths, who didn't *really* know what they were doing.

The major obstacle the experimentalist must navigate around is to make sure they the experiment they are doing is the one they *think* they are doing. Symbolically, when we want to know $\Pr(Y|do(X=x))$, we need to make sure that we are *only* manipulating X, and not accidentally doing $\Pr(Y|do(X=x), Z=z)$ (because we are only experimenting on a sub-population), or $\Pr(Y|do(X=x, Z=z))$ (because we are also, inadvertently, manipulating Z). There are two big main divisions about how to avoid these confusions.

1. The older strategy is to *deliberately* control or manipulate as many other variables as possible. If we find $\Pr(Y|do(X=x, Z=z))$ and $\Pr(Y|do(X=x', Z=z))$ then we know the differences between them are indeed just due to changing X. This strategy, of actually controlling or manipulating whatever we can, is the traditional one in the physical sciences, and more or less goes back to Galileo and the beginning of the Scientific Revolution[1].
2. The younger strategy is to *randomize* over all the other variables but X. That is, to examine the contrast between $\Pr(Y|do(X=x))$ and $\Pr(Y|do(X=x'))$,

[1] The anguished sound you hear as you read this is every historian of science wailing in protest as the over-simplification, but this will do as an origin myth for our purposes.

we use an independent source of random noise to decide which experimental subjects will get $do(X = x)$ and which will get $do(X = x')$. It is easy to convince yourself that this makes $\Pr(Y|do(X = x))$ equal to $\Pr(Y|X = x)$. The great advantage of the randomization approach is that we can apply it even when we cannot actually control the other causally relevant variables, or even are unsure of what they are. Unsurprisingly, it has its origins in the biological sciences, especially agriculture. If we want to credit its invention to a single culture hero, it would not be too misleading[2] to attribute it to R. A. Fisher in the early 1900s.

Experimental evidence is compelling, but experiments are often slow, expensive, and difficult. Moreover, experimenting on people is hard, both because there are many experiments we *shouldn't* do, and because there are many experiments which would just be too hard to organize. We must therefore consider how to do causal inference from non-experimental, observational data.

20.2 Identification and Confounding

For the present purposes, the most important distinction between probabilistic and causal conditioning has to do with the **identification** (or **identifiability**), of the conditional distributions. An aspect of a statistical model is **identifiable** when it cannot be changed without there also being *some* change in the distribution of the observable variables. If we can alter part of a model with no observable consequences, that part of the model is **unidentifiable**[3]. Sometimes the lack of identification is trivial: in a two-cluster mixture model, we get the same observable distribution if we swap the labels of the two clusters (§17.1.5). The rotation problem for factor models (§§16.5, 16.9.1) is a less trivial identification problem[4]. If two variables are co-linear, then their coefficients in a linear regression are unidentifiable (§2.1.1)[5]. Note that identification is about the true distribution, not about what happens with finite data. A parameter might be identifiable, but we could have so little information about it in our data that our estimates are unusable, with immensely wide confidence intervals; that's unfortunate, but we just need more data. An unidentifiable parameter, however, cannot be estimated even with infinite data.[6]

When X and Y are both observable variables, $\Pr(Y|X = x)$ can't help being

[2] See previous note.

[3] More formally, divide the model's parameters into two parts, say θ and ψ. The distinction between θ_1 and θ_2 is identifiable if, for all ψ_1, ψ_2, the distribution over observables coming from (θ_1, ψ_1) is different from that coming from (θ_2, ψ_2). If the right choice of ψ_1 and ψ_2 masks the distinction between θ_1 and θ_2, then θ is unidentifiable.

[4] As this example suggests, what is identifiable depends on what is observed. If we could observe the factors directly, factor loadings would be identifiable.

[5] As that example suggests, whether one aspect of a model is identifiable or not can depend on other aspects of the model. If the co-linearity was broken, the two regression coefficients would become identifiable.

[6] For more on identifiability, and what to do with unidentifiable problems, see the great book by Manski (2007).

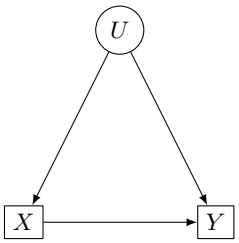


Figure 20.1 The distribution of Y given X, $\Pr(Y|X)$, **confounds** the actual causal effect of X on Y, $\Pr(Y|do(X = x))$, with the indirect dependence between X and Y created by their unobserved common cause U. (You may imagine that U is really more than one variable, with some internal sub-graph.)

identifiable. (Changing this conditional distribution just *is* changing part of the distribution of observables.) Things are very different, however, for $\Pr(Y|do(X = x))$. In some models, it's entirely possible to change this drastically, and always have the same distribution of observables, by making compensating changes to other parts of the model. When this is the case, we simply cannot estimate causal effects from observational data. The basic problem is illustrated in Figure 20.1.

In Figure 20.1, X is a parent of Y. But if we analyze the dependence of Y on X, say in the form of the conditional distribution $\Pr(Y|X = x)$, we see that there are two channels by which information flows from cause to effect. One is the direct, causal path, represented by $\Pr(Y|do(X = x))$. The other is the indirect path, where X gives information about its parent U, and U gives information about its child Y. If we just observe X and Y, we cannot separate the causal effect from the indirect inference. The causal effect is **confounded** with the indirect inference. More generally, the effect of X on Y is confounded whenever $\Pr(Y|do(X = x)) \neq \Pr(Y|X = x)$. If there is some way to write $\Pr(Y|do(X = x))$ in terms of distributions of observables, we say that the confounding can be removed by an **identification strategy**, which **deconfounds** the effect. If there is no way to de-confound, then this causal effect is unidentifiable.

The effect of X on Y in Figure 20.1 is unidentifiable. Even if we erased the arrow from X to Y, we could get any joint distribution for X and Y we liked by picking $P(X|U)$, $P(Y|U)$ and $P(U)$ appropriately. So we cannot even, in this situation, use observations to tell whether X is actually a cause of Y. Notice, however, that even if U was observed, it would still not be the case that $\Pr(Y|X = x) = \Pr(Y|do(X = x))$. While the effect would be identifiable (via the back door criterion; see below, §20.3.1), we would still need some sort of adjustment to recover it.

In the next section, we will look at such identification strategies and adjustments.

20.3 Identification Strategies

To recap, we want to calculate the causal effect of X on Y, $\Pr(Y|do(X=x))$, but we cannot do an experiment, and must rely on observations. In addition to X and Y, there will generally be some **covariates** Z which we know, and we'll assume we know the causal graph, which is a DAG. Is this enough to determine $\Pr(Y|do(X=x))$? That is, does the joint distribution **identify** the causal effect?

The answer is "yes" when the covariates Z contain all the other relevant variables[7]. The inferential problem is then no worse than any other statistical estimation problem. In fact, if we know the causal graph and get to observe all the variables, then we could (in principle) just use our favorite non-parametric conditional density estimate at each node in the graph, with its parent variables as the inputs and its own variable as the response. Multiplying conditional distributions together gives the whole distribution of the graph, and we can get any causal effects we want by surgery. Equivalently (Exercise 20.2), we have that

$$\Pr(Y|do(X=x)) = \sum_t \Pr(Y|X=x, \mathrm{Pa}(X)=t) \Pr(\mathrm{Pa}(X)=t) \quad (20.3)$$

where $\mathrm{Pa}(X)$ is the complete set of parents of X. If we're willing to assume more, we can get away with just using non-parametric regression or even just an additive model at each node. Assuming yet more, we could use parametric models at each node; the linear-Gaussian assumption is (alas) very popular.

If some variables are *not* observed, then the issue of which causal effects are observationally identifiable is considerably trickier. Apparently subtle changes in which variables are available to us and used can have profound consequences.

! The basic principle underlying all considerations is that we would like to condition on adequate **control** variables, which will block paths linking X and Y *other than* those which would exist in the surgically-altered graph where all paths into X have been removed. If other unblocked paths exist, then there is some confounding of the causal effect of X on Y with their mutual dependence on other variables.

This is familiar to use from regression as the basic idea behind using additional variables in our regression, where the idea is that by introducing covariates, we

[7] This condition is sometimes known as **causal sufficiency**. Strictly speaking, we do not have to suppose that *all* causes are included in the model and observable. What we have to assume is that all of the remaining causes have such an unsystematic relationship to the ones included in the DAG that they can be modeled as noise. (This does not mean that the noise is necessarily small.) In fact, what we really have to assume is that the relationships between the causes omitted from the DAG and those included is so intricate and convoluted that it might as well be noise, along the lines of algorithmic information theory (Li and Vitányi, 1997), whose key result might be summed up as "Any determinism distinguishable from randomness is insufficiently complex". But here we verge on philosophy.

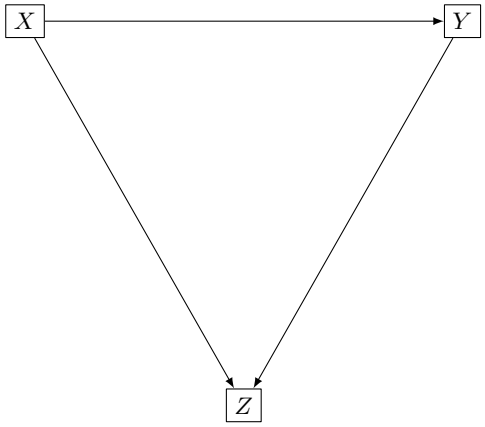


Figure 20.2 "Controlling for" additional variables can introduce bias into estimates of causal effects. Here the effect of X on Y is directly identifiable, $\Pr(Y|do(X = x)) = \Pr(Y|X = x)$. If we also condition on Z however, because it is a common *effect* of X and Y, we'd get $\Pr(Y|X = x, Z = z) \neq \Pr(Y|X = x)$. In fact, even if there were no arrow from X to Y, conditioning on Z would make Y depend on X.

"control for" other effects, until the regression coefficient for our favorite variable represents only its causal effect. Leaving aside the inadequacies of linear regression as such (Chapter 2), we need to be cautious here. Just conditioning on everything possible does *not* give us adequate control, or even necessarily bring us closer to it. As Figure 20.2 illustrates, and as several of the data-analysis problem sets will drive home [[CROSS-REF]], *adding* an ill-chosen covariate to a regression can create confounding.

There are three main ways we can find adequate controls, and so get both identifiability and appropriate adjustments:

1. We can condition on an intelligently-chosen set of covariates S, which block all the indirect paths from X to Y, but leave all the direct paths open. (That is, we can follow the regression strategy, but do it right.) To see whether a candidate set of controls S is adequate, we apply the **back-door criterion**.
2. We can find a set of variables M which **mediate** the causal influence of X on Y — all of the direct paths from X to Y pass through M. If we can identify the effect of M on Y, and of X on M, then we can combine these to get the effect of X on Y. (That is, we can just study the *mechanisms* by which X influences Y.) The test for whether we can do this combination is the **front-door criterion**.
3. We can find a variable I which affects X, and which *only* affects Y by influ-

encing X. If we can identify the effect of I on Y, and of I on X, then we can, sometimes, "factor" them to get the effect of X on Y. (That is, I gives us variation in X which is independent of the common causes of X and Y.) I is then an **instrumental variable** for the effect of X on Y.

Let's look at these three in turn.

20.3.1 The Back-Door Criterion: Identification by Conditioning

When estimating the effect of X on Y, a **back-door path** is an undirected path between X and Y with an arrow *into* X. These are the paths which create confounding, by providing an indirect, non-causal channel along which information can flow. A set of conditioning variables or controls S satisfies the **back-door criterion** when (i) S blocks every back-door path between X and Y, and (ii) no node in S is a descendant of X. (Cf. Figure 20.3.) When S meets the back-door criterion,

$$\Pr(Y|do(X=x)) = \sum_s \Pr(Y|X=x, S=s)\Pr(S=s) \tag{20.4}$$

Notice that all the items on the right-hand side are observational conditional probabilities, not counterfactuals. Thus we have achieved identifiability, as well as having an adjustment strategy.

The motive for (i) is plain, but what about (ii)? We don't want to include descendants of X which are also ancestors of Y, because that blocks off some of the causal paths from X to Y, and we don't want to include descendants of X which are also descendants of Y, because they provide non-causal information about Y[8].

More formally, we can proceed as follows (Pearl, 2009b, §11.3.3). We know from Eq. 20.3 that

$$\Pr(Y|do(X=x)) = \sum_t \Pr(\mathrm{Pa}(X)=t)\Pr(Y|X=x, \mathrm{Pa}(X)=t) \tag{20.5}$$

We can always introduce another set of conditioned variables, if we also sum over them:

$$\Pr(Y|do(X=x)) = \sum_t \Pr(\mathrm{Pa}(X)=t)\sum_s \Pr(Y, S=s|X=x, \mathrm{Pa}(X)=t) \tag{20.6}$$

We can do this for *any* set of variables S, it's just probability. It's also just probability that

$$\begin{aligned}&\Pr(Y, S|X=x, \mathrm{Pa}(X)=t) = \\ &\quad \Pr(Y|X=x, \mathrm{Pa}(X)=t, S=s)\Pr(S=s|X=x, \mathrm{Pa}(X)=t)\end{aligned} \tag{20.7}$$

[8] What about descendants of X which are neither ancestors nor descendants of Y? Conditioning on them is either creates potential colliders, if they are also descended from ancestors of Y other than X, or needlessly complicates the adjustment in Eq. 20.4.

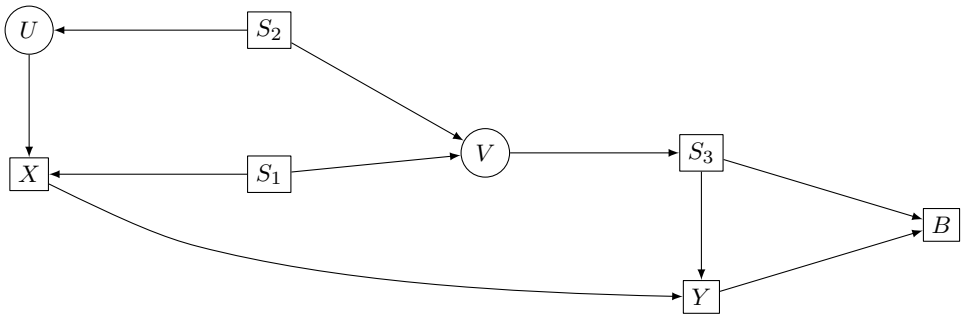


Figure 20.3 Illustration of the back-door criterion for identifying the causal effect of X on Y. Setting $S = \{S_1, S_2\}$ satisfies the criterion, but neither S_1 nor S_2 on their own would. Setting $S = \{S_3\}$, or $S = \{S_1, S_2, S_3\}$ also works. Adding B to any of the good sets makes them fail the criterion.

so

$$\Pr(Y|do(X=x)) = \sum_t \Pr(\mathrm{Pa}(X)=t) \sum_s \Pr(Y|X=x, \mathrm{Pa}(X)=t, S=s) \Pr(S=s|X=x, \mathrm{Pa}(X)=t) \tag{20.8}$$

Now we use the fact that S satisfies the back-door criterion. Point (i) of the criterion, blocking back-door paths, implies that $Y \perp\!\!\!\perp \mathrm{Pa}(X)|X, S$. Thus

$$\Pr(Y|do(X=x)) = \sum_t \Pr(\mathrm{Pa}(X)=t) \sum_s \Pr(Y|X=x, S=s) \Pr(S=s|X=x, \mathrm{Pa}(X)=t) \tag{20.9}$$

Point (ii) of the criterion, not containing descendants of X, means (by the Markov property) that $X \perp\!\!\!\perp S|\mathrm{Pa}(X)$. Therefore

$$\Pr(Y|do(X=x)) = \sum_t \Pr(\mathrm{Pa}(X)=t) \sum_s \Pr(Y|X=x, S=s) \Pr(S=s|\mathrm{Pa}(X)=t) \tag{20.10}$$

Since $\sum_t \Pr(\mathrm{Pa}(X)=t) \Pr(S=s|\mathrm{Pa}(X)=t) = \Pr(S=s)$, we have, at last,

$$\Pr(Y|do(X=x)) = \sum_s \Pr(Y|X=x, S=s) \Pr(S=s) \tag{20.11}$$

as promised. □

20.3.1.1 The Entner Rules

Using the back-door criterion requires us to know the causal graph. Recently, Entner *et al.* (2013) have given a set of rules which provide *sufficient* conditions

for deciding that set of variables satisfy the back-door criterion, or that X actually has no effect on Y, which can be used without knowing the graph completely.

It makes no sense to control for anything which is a descendant of either Y or X; that's either blocking a directed path, or activating a collider, or just irrelevant. So let $\mathcal{W}$ be the set of all observed variables which descend neither from X nor Y.

1. If there is a set of controls S such that $X \perp\!\!\!\perp Y|S$, then X has no causal effect on Y.

   Reasoning: Y can't be a child of X if we can make them independent by conditioning on anything, and Y can't be a more remote descendant either, since S doesn't include any descendants of X. So in this situation all the paths linking X to Y must be back-door paths, and S, blocking them, shows there's no effect.
2. If there is a $W \in \mathcal{W}$ and a subset S of the $\mathcal{W}$, not including W, such that (i) $W \not\perp\!\!\!\perp Y|S$, but (ii) $W \perp\!\!\!\perp Y|S, X$, then X has an effect on Y, and S satisfies the back-door criterion for estimating the effect.

   Reasoning: Point (i) shows that conditioning on S leaves open path from W to Y. By point (ii), these paths must all pass through X, since conditioning on X blocks them, hence X has an effect on Y. S must block all the back-door paths between X and Y, otherwise X would be a collider on paths between W and Y, so conditioning on X would activate those paths.
3. If there is a $W \in \mathcal{W}$ and a subset S of $\mathcal{W}$, excluding W, such that (i) $W \not\perp\!\!\!\perp X|S$ but (ii) $W \perp\!\!\!\perp Y|S$, then X has no effect on Y.

   Reasoning: Point (i) shows that conditioning on S leaves open active paths from W to X. But by (ii), there cannot be any open paths from W to Y, so there cannot be any open paths from X to Y.

If none of these rules apply, whether X has an effect on Y, and if so what adequate controls are for finding it, will depend on the exact graph, and *cannot* be determined just from independence relations among the observables. (For proofs of everything, see the paper.)

20.3.2 The Front-Door Criterion: Identification by Mechanisms

A set of variables M satisfies the **front-door criterion** when (i) M blocks all directed paths from X to Y, (ii) there are no unblocked back-door paths from X to M, and (iii) X blocks all back-door paths from M to Y. Then

$$\Pr(Y|do(X=x)) = \sum_m \Pr(M=m|X=x) \sum_{x'} \Pr(Y|X=x', M=m) \Pr(X=x') \tag{20.12}$$

The variables M are sometimes called **mediators**.

A natural reaction to the front-door criterion is "Say what?", but it becomes more comprehensible if we take it apart. Because, by clause (i), M blocks all

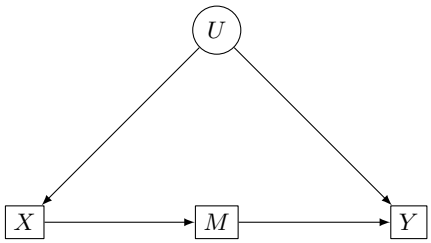


Figure 20.4 Illustration of the front-door criterion, after Pearl (2009b, Figure 3.5). X, Y and M are all observed, but U is an unobserved common cause of both X and Y. $X \leftarrow U \rightarrow Y$ is a back-door path confounding the effect of X on Y with their common cause. However, all of the effect of X on Y is mediated through X's effect on M. M's effect on Y is, in turn, confounded by the back-door path $M \leftarrow X \leftarrow U \rightarrow Y$, but X blocks this path. So we can use back-door adjustment to find $\Pr(Y|do(M=m))$, and directly find $\Pr(M|do(X=x)) = \Pr(M|X=x)$. Putting these together gives $\Pr(Y|do(X=x))$.

directed paths from X to Y, any *causal* dependence of Y on X must be mediated by a dependence of Y on M:

$$\Pr(Y|do(X=x)) = \sum_m \Pr(Y|do(M=m)) \Pr(M=m|do(X=x)) \quad (20.13)$$

Clause (ii) says that we can get the effect of X on M directly,

$$\Pr(M=m|do(X=x)) = \Pr(M=m|X=x) \ . \quad (20.14)$$

Clause (iii) say that X satisfies the back-door criterion for identifying the effect of M on Y, and the inner sum in Eq. 20.12 is just the back-door computation (Eq. 20.4) of $\Pr(Y|do(M=m))$. So really we *are* using the back door criterion, twice. (See Figure 20.4.)

For example, in the "does tooth-brushing prevent heart-disease?" example of §19.2.2, we have X = "frequency of tooth-brushing", Y = "heart disease", and we could take as the mediating M either "gum disease" or "inflammatory immune response", according to Figure 19.2.

20.3.2.1 The Front-Door Criterion and Mechanistic Explanation

Morgan and Winship (2007, ch. 8) give a useful insight into the front-door criterion. Each directed path from X to Y is, or can be thought of as, a separate **mechanism** by which X influences Y. The requirement that all such paths be blocked by M, (i), is the requirement that the set of mechanisms included in M be "exhaustive". The two back-door conditions, (ii) and (iii), require that the mechanisms be "isolated", not interfered with by the rest of the data-generating process (at least once we condition on X). Once we identify an isolated and ex-

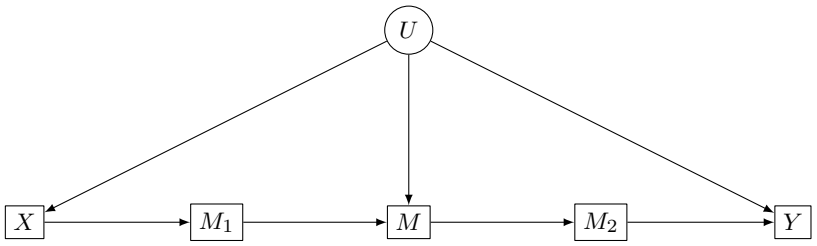


Figure 20.5 The path $X \to M \to Y$ contains all the mechanisms by which X influences Y, but is not isolated from the rest of the system ($U \to M$). The sub-mechanisms $X \to M_1 \to M$ and $M \to M_2 \to Y$ *are* isolated, and the original causal effect can be identified by composing them.

haustive set of mechanisms, we know all the ways in which X actually affects Y, and any indirect paths can be discounted, using the front-door adjustment 20.12.

One interesting possibility suggested by this is to elaborate mechanisms into sub-mechanisms, which could be used in some cases where the plain front-door criterion won't apply[9], such as Figure 20.5. Because U is a parent of M, we cannot use the front-door criterion to identify the effect of X on Y. (Clause (i) holds, but (ii) and (iii) both fail.) But we can use M_1 and the front-door criterion to find $\Pr(M|do(X = x))$, and we can use M_2 to find $\Pr(Y|do(M = m))$. Chaining those together, as in Eq. 20.13, would given $\Pr(Y|do(X = x))$. So even though the whole mechanism from X to Y is not isolated, we can still identify effects by breaking it into sub-mechanisms which *are* isolated. This suggests a natural point at which to stop refining our account of the mechanism into sub-sub-sub-mechanisms: when we can identify the causal effects we're concerned with.

20.3.3 Instrumental Variables

A variable I is an **instrument**[10] for identifying the effect of X on Y when there is a set of controls S such that (i) $I \not\perp\!\!\!\perp X|S$, and (ii) every unblocked path from I to Y has an arrow pointing into X. Another way to say (ii) is that $I \perp\!\!\!\perp Y|S, do(X)$. Colloquially, I influences Y, but only through first influencing X (at least once we control for S). (See Figure 20.6.)

How is this useful? By making back-door adjustments for S, we can identify $\Pr(Y|do(I = i))$ and $\Pr(X|do(I = i))$. Since all the causal influence of I on Y

[[TODO: Correspondence with D. Blei and E. Oblander shows me that some of what's being done here to show identification really needs extra assumptions; revise (and consult Singh *et al.* (2019) in doing so)]]

[9] The ideas in this paragraph come from conversation Prof. Winship; see Morgan and Winship (2015, ch. 10).

[10] The term "instrumental variables" comes from econometrics, where they were originally used, in the 1940s, to identify parameters in simultaneous equation models. (The metaphor was that I is a measuring instrument for the otherwise inaccessible parameters.) Definitions of instrumental variables are surprisingly murky and controversial outside of extremely simple linear systems; this one is taken from Galles and Pearl (1997), via Pearl (2009b, §7.4.5).

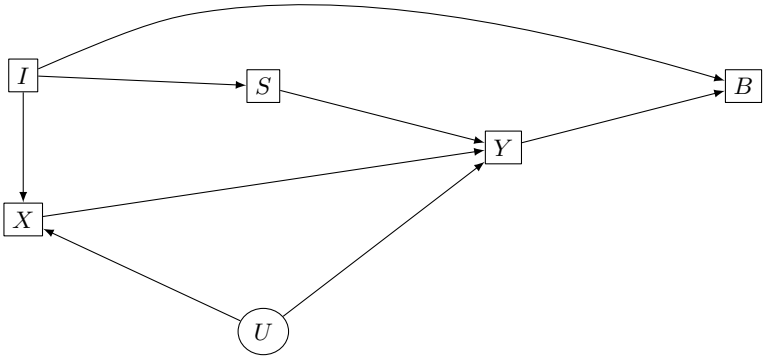


Figure 20.6 A valid instrumental variable, I, is related to the cause of interest, X, and influences Y only through its influence on X, at least once control variables block other paths. Here, to use I as an instrument, we *should* condition on S, but *should not* condition on B. (If we could condition on U, we would not need to use an instrument.)

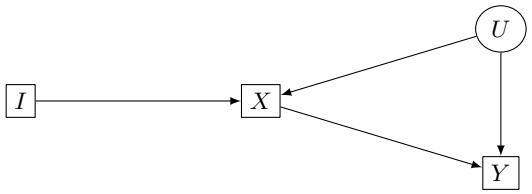


Figure 20.7 I acts as an instrument for estimating the effect of X on Y, despite the presence of the confounding, unobserved variable U.

must be channeled through X (by point (ii)), we have

$$\Pr(Y|do(I=i)) = \sum_x \Pr(Y|do(X=x)) \Pr(X=x|do(I=i)) \tag{20.15}$$

as in Eq. 20.3. We can thus identify the causal effect of X on Y whenever Eq. 20.15 can be solved for $\Pr(Y|do(X=x))$ in terms of $\Pr(Y|do(I=i))$ and $\Pr(X|do(I=i))$. Figuring out when this is possible in general requires an excursion into the theory of integral equations[11], which I have bracketed in §20.3.3.3. The upshot is that while there *may* not be unique solutions, there *often* are, though they can be somewhat hard to calculate. However, in the special case where the relations between all variables are linear, we can be much more specific, fairly easily.

Let's start with the most basic possible set-up for an instrumental variable, namely that in Figure 20.7, where we just have X, Y, the instrument I, and the

[11] If X is continuous, then the analog of Eq. 20.15 is $\Pr(Y|do(I=i)) = \int p(Y|do(X=x))p(X=x|do(I=i))dx$, where the "integral operator" $\int \cdot p(X=x|do(I=i))dx$ is known, as is $\Pr(Y|do(I=i))$.

unobserved confounders S. If everything is linear, identifying the causal effect of X on Y is equivalent to identifying the coefficient on the $X \to Y$ arrow. We can write

$$X = \alpha_0 + \alpha I + \delta U + \epsilon_X \tag{20.16}$$

and

$$Y = \beta_0 + \beta X + \gamma U + \epsilon_Y \tag{20.17}$$

where ϵ_X and ϵ_Y are mean-zero noise terms, independent of each other and of the other variables, and we can, without loss of generality, assume U has mean zero as well. We want to find β. Substituting,

$$Y = \beta_0 + \beta\alpha_0 + \beta\alpha I + (\beta\delta + \gamma)U + \beta\epsilon_X + \epsilon_Y \tag{20.18}$$

Since U, ϵ_X and ϵ_Y are all unobserved, we can re-write this as

$$Y = \gamma_0 + \beta\alpha I + \eta \tag{20.19}$$

where $\gamma_0 = \beta_0 + \beta\alpha_0$, and $\eta = (\beta\delta + \gamma)U + \beta\epsilon_X + \epsilon_Y$ has mean zero.

Now take the covariances:

$$\begin{aligned}
\mathrm{Cov}\left[I, X\right] &= \alpha\mathbb{V}\left[I\right] + \mathrm{Cov}\left[\epsilon_X, I\right] && (20.20)\\
\mathrm{Cov}\left[I, Y\right] &= \beta\alpha\mathbb{V}\left[I\right] + \mathrm{Cov}\left[\eta, I\right] && (20.21)\\
&= \beta\alpha\mathbb{V}\left[I\right] + (\beta\delta + \gamma)\mathrm{Cov}\left[U, I\right] && (20.22)\\
&\quad + \beta\mathrm{Cov}\left[\epsilon_X, I\right] + \mathrm{Cov}\left[\epsilon_Y, I\right]
\end{aligned}$$

By condition (ii), however, we must have $\mathrm{Cov}\left[U, I\right] = 0$, and of course $\mathrm{Cov}\left[\epsilon_X, I\right] = \mathrm{Cov}\left[\epsilon_Y, I\right] = 0$. Therefore $\mathrm{Cov}\left[I, Y\right] = \beta\alpha\mathbb{V}\left[I\right]$. Solving,

$$\beta = \frac{\mathrm{Cov}\left[I, Y\right]}{\mathrm{Cov}\left[I, X\right]} \tag{20.23}$$

This can be estimated by substituting in the sample covariances, or any other consistent estimators of these two covariances. (§21.2 covers IV estimation in more detail.)

On the other hand, the (true or population-level) coefficient for linearly regressing Y on X is

$$\begin{aligned}
\frac{\mathrm{Cov}\left[X, Y\right]}{\mathbb{V}\left[X\right]} &= \frac{\beta\mathbb{V}\left[X\right] + \gamma\mathrm{Cov}\left[U, X\right]}{\mathbb{V}\left[X\right]} && (20.24)\\
&= \beta + \gamma\frac{\mathrm{Cov}\left[U, X\right]}{\mathbb{V}\left[X\right]} && (20.25)\\
&= \beta + \gamma\frac{\delta\mathbb{V}\left[U\right]}{\alpha^2\mathbb{V}\left[I\right] + \delta^2\mathbb{V}\left[U\right] + \mathbb{V}\left[\epsilon_X\right]} && (20.26)
\end{aligned}$$

That is, "OLS is biased for the causal effect when X is correlated with the noise". In other words, simple regression is misleading in the presence of confounding[12].

[12] But observe that if we want to make a linear prediction of Y and only have X available, i.e., to find the best r_1 in $\mathbb{E}\left[Y|X = x\right] = r_0 + r_1 x$, then Eq. 20.26 is *exactly* the coefficient we would want to use. OLS is doing its job.

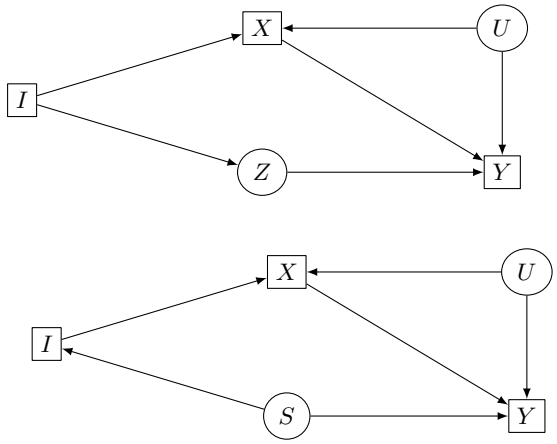


Figure 20.8 Left: I is not a valid instrument for identifying the effect of X on Y, because I can influence Y through a path not going through X. If we could control for Z, however, I would become valid. Right: I is not a valid instrument for identifying the effect of X on Y, because there is an unblocked back-door path connecting I and Y. If we could control for S, however, I would become valid.

The instrumental variable I provides a source of variation in X which is uncorrelated with the other common ancestors of X and Y. By seeing how both X and Y respond to these perturbations, and using the fact that I only influences Y through X, we can deduce something about how X influences Y, though linearity is very important to our ability to do so.

The simple line of reasoning above runs into trouble if we have multiple instruments, or need to include controls (as the definition of an instrument allows). §21.2 will also look at the more complicated estimation methods which can handle this, again assuming linearity.

20.3.3.1 Some Invalid Instruments

Not everything which looks like an instrument actually works. The ones which don't are called **invalid** instruments. If Y is indeed a descendant of I, but there is a line of descent that doesn't go through X, then I is not a valid instrument for X (Figure 20.8, left). If there are unblocked back-door paths linking I and Y, e.g., if I and Y have common ancestors, then I is again not a valid instrument (Figure 20.8, right).

Economists sometimes refer to both sets of problems with instruments as "violations of exclusion restrictions". The second sort of problem, in particular, is a "failure of exogeneity".

20.3.3.2 Critique of Instrumental Variables

By this point, you may well be thinking that instrumental variable estimation is very much like using the front-door criterion. There, the extra variable M came between X and Y; here, X comes between I and Y. It is, perhaps, surprising (if not annoying) that using an instrument only lets us identify causal effects under extra assumptions, but that's (mathematical) life. Just as the front-door criterion relies on using our scientific knowledge, or rather theories, to find isolated and exhaustive mechanisms, finding valid instruments relies on theories about the part of the world under investigation, and one would want to try to check those theories.

In fact, instrumental variable estimates of causal effects are often presented as more or less unquestionable, and free of theoretical assumptions; economists, and other social scientists influenced by them, are especially apt to do this. As the economist Daniel Davies puts it[13], devotees of this approach

> have a really bad habit of saying:
> "Whichever way you look at the numbers, X".
> when all they can really justify is:
> "Whichever way **I** look at the numbers, X".
> but in fact, I should have said that they could only really support:
> "Whichever way **I** look at **these** numbers, X".

(Emphasis in the original.) It will not surprise you to learn that I think this is very wrong.

I hope that, by this point in the book, if someone tries to sell you a linear regression, you should be very skeptical, but let's leave that to one side. (It's *possible* that the problem at hand really is linear.) The clue that instrumental variable estimation is a creature of theoretical assumptions is point (ii) in the definition of an instrument: $I \perp\!\!\!\perp Y|S, do(X)$. This says that if we eliminate all the arrows into X, the control variables S block all the other paths between I and Y. This is *exactly* as much an assertion about mechanisms as what we have to do with the front-door criterion. In fact it doesn't just say that every mechanism by which I influences Y is mediated by X, it also says that there are no common causes of I and Y (other than those blocked by S).

This assumption is most easily defended when I is genuinely random, For instance, if we do a randomized experiment, I might be a coin-toss which assigns each subject to be in either the treatment or control group, each with a different value of X. If "compliance" is not perfect (if some of those in the treatment group don't actually get the treatment, or some in the control group do), it is nonetheless often plausible that the only route by which I influences the outcome is through X, so an instrumental variable regression is appropriate. (I here is sometimes called "intent to treat".)

Even here, we must be careful. If we are evaluating a new medicine, whether people *think* they are getting a medicine or not could change how they act, and

[13] In part four of his epic and insightful review of *Freakonomics*; see `http://d-squareddigest.blogspot.com/2007/09/freakiology-yes-folks-its-part-4-of.html`.

medical outcomes. Knowing whether they were assigned to the treatment or the control group would thus create another path from I to Y, not going through X. This is why randomized clinical trials are generally "double-blinded" (neither patients nor medical personnel know who is in the control group); but whether the steps taken to double-blind the trial actually worked is itself a causal assumption.

More generally, any argument that a candidate instrument is valid is really an argument that other channels of information flow, apart from the favored one through X, can be ruled out. This generally cannot be done through analyzing the same variables used in the instrumental-variable estimation (see below), but involves theories about the world, and rests on the strength of the evidence for those theories. As has been pointed out multiple times — e.g., by Rosenzweig and Wolpin (2000) and Deaton (2010) — the theories needed to support instrumental variable estimates in particular concrete cases are often *not* very well-supported, and plausible rival theories can produce very different conclusions from the same data.

Many people have thought that one *can* test for the validity of an instrument, by looking at whether $I \perp\!\!\!\perp Y|X$ — the idea being that, if influence flows from I through X to Y, conditioning on X should block the channel. The problem is that, in the instrumental-variable set-up, X is a collider on the path $I \rightarrow X \leftarrow U \rightarrow Y$, so conditioning on X actually creates an indirect dependence between I and Y *even if* I is valid. So $I \not\perp\!\!\!\perp Y|X$, whether or not the instrument is valid, and the test (even if done perfectly with infinite data) tells us nothing[14].

A final, more or less technical, issue with instrumental variable estimation is that many instruments are (even if valid) **weak** — they only have a little influence on X, and a small covariance with it. This means that the denominator in Eq. 20.23 is a number close to zero. Error in estimating the denominator, then, results in a much larger error in estimating the ratio. Weak instruments lead to noisy and imprecise estimates of causal effects (§**??**). It is not hard to construct scenarios where, at reasonable sample sizes, one is actually better off using the biased OLS estimate than the unbiased but high-variance instrumental estimate[15].

20.3.3.3 Instrumental Variables and Integral Equations

I said above (p. 20.3.3) that, in general, identifying causal effects through instrumental variables means solving integral equations. It's worth exploring that, because it provides some insight into how instrumental variables works, especially for non-linear systems. Since this is somewhat mathematically involved, however, you may want to skip this section on first reading.

To grasp what it means to identify causal effects by solving integral equations, let's start with the most basic set up, where the cause X, the effect Y, and the instrument I are all binary. There are then really only two numbers that

[14] However, see Pearl (2009b, §8.4) for a different approach which can "screen out very bad would-be instruments".

[15] Young (2017) re-analyzes hundreds of published papers in economics to argue that this is scenario is actually rather common.

need to be identified, $\Pr(Y = 1|do(X = 0))$ and $\Pr(Y = 1|do(X = 1))$. Eq. 20.15 becomes now a system of equations involving these effects:

$$\Pr(Y = 1|do(I = 0)) = \Pr(Y = 1|do(X = 0)) \Pr(X = 0|do(I = 0)) + \Pr(Y = 1|do(X = 1)) \Pr(X =$$
$$\Pr(Y = 1|do(I = 0)) = \Pr(Y = 1|do(X = 0)) \Pr(X = 0|do(I = 1)) + \Pr(Y = 1|do(X = 1)) \Pr(X =$$

The left-hand sides are identifiable (by the assumptions on I), as are the probabilities $\Pr(X|do(I))$. So, once we get those, we have a system of two linear equations with two unknowns, $\Pr(Y = 1|do(X = 0))$ and $\Pr(Y = 1|do(X = 1))$. Since there are as many equations as unknowns, there is a unique solution, unless the equations are redundant (Exercise 20.4).

If we put together some vectors and matrices,

$$\vec{f}_{I\to Y} \equiv \begin{bmatrix} \Pr(Y = 1|do(I = 0)) \\ \Pr(Y = 1|do(I = i)) \end{bmatrix} \tag{20.28}$$

$$\vec{f}_{X\to Y} \equiv \begin{bmatrix} \Pr(Y = 1|do(X = 0)) \\ \Pr(Y = 1|do(X = 1)) \end{bmatrix} \tag{20.29}$$

$$\mathbf{f}_{I\to X} \equiv \begin{bmatrix} \Pr(X = 0|do(I = 0)) & \Pr(X = 1|do(I = 0)) \\ \Pr(X = 0|do(I = 1)) & \Pr(X = 1|do(I = 1)) \end{bmatrix} \tag{20.30}$$

then Eq. 20.27 becomes

$$\vec{f}_{I\to Y} = \mathbf{f}_{I\to X}\vec{f}_{X\to Y} \tag{20.31}$$

and we can make the following observations:

1. The effect of the instrument I on the response Y, $\vec{f}_{I\to Y}$, is a linear transformation of the desired causal effects, $\vec{f}_{X\to Y}$.
2. Getting those desired effects requires inverting a linear operator, the matrix $\mathbf{f}_{I\to X}$.
3. That inversion is possible if, and only if, all of the eigenvalues of $\mathbf{f}_{I\to X}$ are non-zero.

There is nothing too special about the all-binary case, except that we can write everything out explicitly. If the cause, effect and instrument are all categorical, with the number of levels being c_x, c_y and c_i respectively, then there are $(c_y - 1)c_x$ parameters to identify, and Eq. 20.15 leads to a system of $(c_y - 1)c_i$ equations, so the effects will be identifiable (in general) so long as $c_i \geq c_x$. There will, once again, be a matrix form of the system of equations, and solving the system means inverting a matrix in whose entries are the effects of I on X, $\Pr(X = x|do(I = i))$. This, in turn, is something we can do so long as all of the eigenvalues are non-zero.

In the continuous case, we will replace our vectors by conditional density functions:

$$f_{I\to Y}(y|i) \equiv f(y|do(I = i)) \tag{20.32}$$
$$f_{X\to Y}(y|x) \equiv f(y|do(X = x)) \tag{20.33}$$
$$f_{I\to X}(x|i) \equiv f(x|do(I = i)) \tag{20.34}$$

Eq. 20.15 now reads

$$f_{I\to Y}(y|i) = \int f_{X\to Y}(y|x) f_{I\to X}(x|i) dx \tag{20.35}$$

This is linear in the desired function $f_{X\to Y}$, so we define the linear operator

$$\Phi h \equiv \int h(x) f_{I\to X}(x|i) dx \tag{20.36}$$

and re-write Eq. 20.15 one last time as

$$f_{I\to Y} = \Phi f_{I\to X} \tag{20.37}$$

which could be solved by

$$\Phi^{-1} f_{I\to Y} = f_{I\to X} \tag{20.38}$$

An operator like Φ is called an "integral operator", and equations like Eq. 20.35 or 20.37 are "integral equations".

If we take Eq. 20.15, multiply both sides by y, and sum (or integrate) over all possible y, we get

$$\mathbb{E}[Y|do(I=i)] = \sum_y \sum_x y \Pr(Y=y|do(X=x)) \Pr(X=x|do(I=i)) \tag{20.39}$$

$$= \sum_x \sum_y y \Pr(Y=y|do(X=x)) \Pr(X=x|do(I=i)) \tag{20.40}$$

$$= \sum_x \Pr(X=x|do(I=i)) \mathbb{E}[Y|do(X=x)] \tag{20.41}$$

$$= \Phi \mathbb{E}[Y|do(X)] \tag{20.42}$$

So, again, the conditional expectations (= average causal effects) we'd like to identify can be obtained by solving a linear integral equation. This doesn't require that either the functions $\mathbb{E}[Y|do(I=i)]$ or $\mathbb{E}[Y|do(X=x)]$ be linear (in i and x, respectively), it just follows from the Markov property[16].

20.3.4 Failures of Identification

The back-door and front-door criteria, and instrumental variables, are all *sufficient* for estimating causal effects from probabilistic distributions, but are not *necessary*. A necessary condition for *un*-identifiability is the presence of an unblockable back-door path from X to Y. However, this is not sufficient for lack of identification — we might, for instance, be able to use the front door criterion, as in Figure 20.4. There are necessary and sufficient conditions for the identifiability of causal effects in terms of the graph, and so for un-identifiability, but they are rather complex and I will not go over them (see Shpitser and Pearl (2008), and Pearl (2009b, §§3.4–3.5) for an overview).

[16] In fact, one reason the Markov property is important in studying dynamics is that it lets us move from studying non-linear individual trajectories to the linear evolution of probability distributions (Lasota and Mackey, 1994).

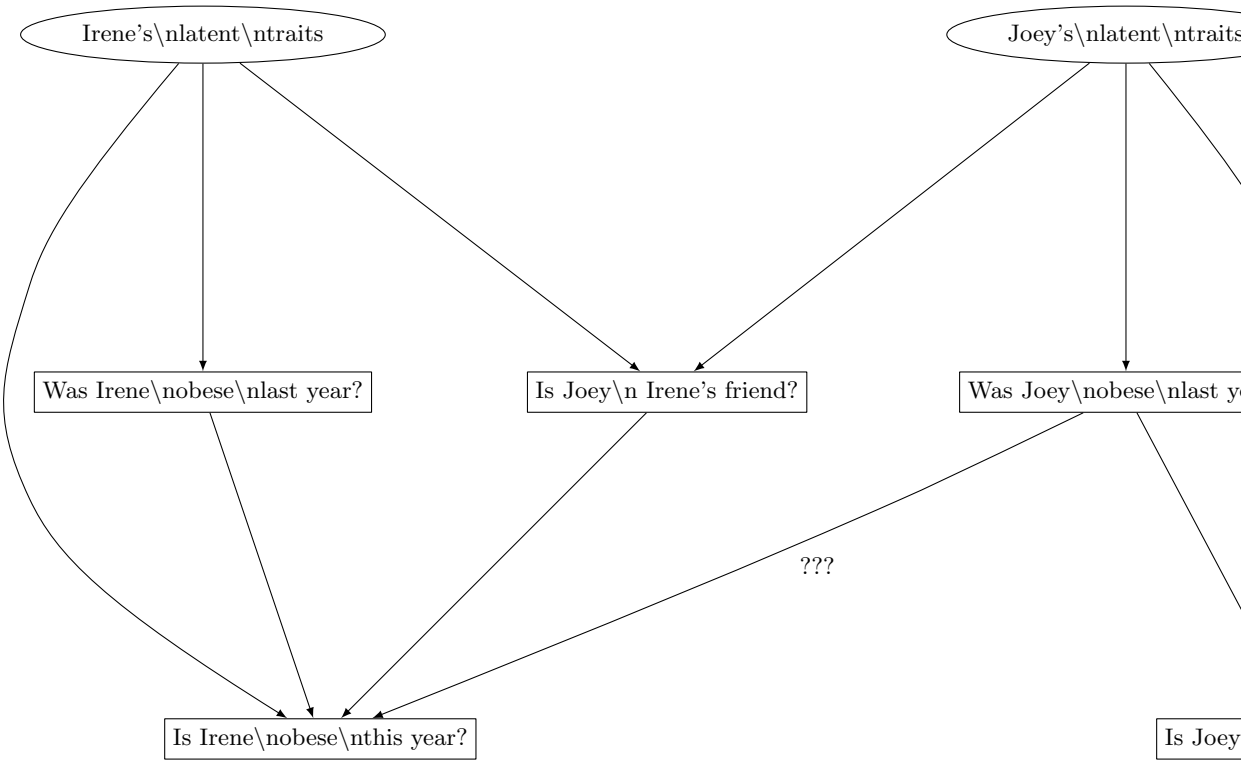


Figure 20.9 Social influence is confounded with selecting friends with similar traits, unobserved in the data.

As an example of the unidentifiable case, consider Figure 20.9. This DAG depicts the situation analyzed in Christakis and Fowler (2007), a famous paper claiming to show that obesity is contagious in social networks (at least in the suburb of Boston where the data was collected). At each observation, participants in the study get their weight taken, and so their obesity status is known over time. They also provide the name of a friend. This friend is often in the study. Christakis and Fowler were interested in the possibility that obesity is contagious, perhaps through some process of behavioral influence. If this is so, then Irene's obesity status in year 2 should depend on Joey's obesity status in year one, but *only* if Irene and Joey are friends — not if they are just random, unconnected people. It is indeed the case that if Joey becomes obese, this predicts a substantial increase in the odds of Joey's friend Irene becoming obese, even controlling for Irene's previous history of obesity[17].

The difficulty arises from the latent variables for Irene and Joey (the round nodes in Figure 20.9). These include all the traits of either person which (a) influence who they become friends with, and (b) influence whether or not they

[17] The actual analysis was a bit more convoluted than that, but this is the general idea.

become obese. A very partial list of these would include: taste for recreational exercise, opportunity for recreational exercise, taste for alcohol, ability to consume alcohol, tastes in food, occupation and how physically demanding it is, ethnic background[18], etc. Put simply, if Irene and Joey are friends because they spend two hours in the same bar every day drinking and eating fried chicken wings with ranch dressing, it's less surprising that both of them have an elevated chance of becoming obese, and likewise if they became friends because they both belong to the decathlete's club, they are both unusually unlikely to become obese. Irene's status is predictable from Joey's, then, not (or not just) because Joey influences Irene, but because seeing what kind of person Irene's friends are tells us about what kind of person Irene is. It is not too hard to convince oneself that there is just no way, in this DAG, to get at the causal effect of Joey's behavior on Irene's that isn't confounded with their latent traits (Shalizi and Thomas, 2011). To de-confound, we would need to actual measure those latent traits, which may not be impossible but is certainly was not done here[19].

When identification is not possible — when we can't de-confound — it may still be possible to *bound* causal effects. That is, even if we can't say exactly that $\Pr(Y|do(X = x))$ must be, we can still say it has to fall within a certain (non-trivial!) range of possibilities. The development of bounds for non-identifiable quantities, what's sometimes called **partial identification**, is an active area of research, which I think is very likely to become more and more important in data analysis; the best introduction I know is Manski (2007).

20.4 Summary

Of the four techniques I have introduced, instrumental variables are clever, but fragile and over-sold[20]. Experimentation is ideal, but often unavailable. The back-door and front-door criteria are, I think, the best observational approaches, when they can be made to work.

Often, nothing can be made to work. Many interesting causal effects are just not identifiable from observational data. More exactly, they only become identifiable under very strong modeling assumptions, typically ones which cannot be tested from the same data, and sometimes ones which cannot be tested by any sort of empirical data whatsoever. Sometimes, we have good reasons (from other parts of our scientific knowledge) to make such assumptions. Sometimes, we make such assumptions because we have a pressing need for *some* basis on which to act, and

[18] Friendships often run within ethnic communities. On the one hand, this means that friends tend to be more *genetically* similar than random members of the same town, so they will be usually apt to share genes which, *in that environment*, influence susceptibility to obesity. On the other hand, ethnic communities transmit, non-genetically, traditions regarding food, alcohol, sports, exercise, etc., and (again non-genetically: Tilly (1998)) influence employment and housing opportunities.

[19] Of course, the issue is not just about obesity. Studies of "viral marketing", and of social influence more broadly, all generically have the same problem. Predicting someone's behavior from that of their friend means conditioning on the existence of a social tie between them, but that social tie is a collider, and activating the collider creates confounding.

[20] I would probably not be so down on them if others did not push them up so excessively.

a wrong guess seems better than nothing[21]. If you do make such assumptions, you need to make clear that you are doing so, and what they are; explain your reasons for making those assumptions, and not others[22]; and indicate how different your conclusions could be if you made different assumptions.

20.4.1 Further Reading

My presentation of the three major criteria is heavily indebted to Morgan and Winship (2007), but I hope not a complete rip-off. Pearl (2009b) is also essential reading on this topic. Berk (2004) provides an excellent critique of naive (that is, overwhelmingly common) uses of linear regression for estimating causal effects .

Most econometrics texts devote considerable space to instrumental variables. Didelez *et al.* (2010) is a very good discussion of instrumental variable methods, with less-standard applications. There is some work on non-parametric versions of instrumental variables (e.g., Newey and Powell 2003), but the form of the models must be restricted or they are unidentifiable. On the limitations of instrumental variables, Rosenzweig and Wolpin (2000) and Deaton (2010) are particularly recommended; the latter reviews the issue in connection with important recent work in development economics and the alleviation of extreme poverty, an area where statistical estimates really do matter.

There is a large literature in the philosophy of science and in methodology on the notion of "mechanisms". References I have found useful include, in general, Salmon (1984), and, specifically on social processes, Elster (1989), Hedström and Swedberg (1998) (especially Boudon 1998), Hedström (2005), Tilly (1984, 2008), and DeLanda (2006).

Exercises

20.1 Draw a graphical model representing the situation where a causal variable X is randomized by an experimenter. Verify that $\Pr(Y|X=x)$ is then equal to $\Pr(Y|do(X=x))$. (*Hint:* Use the back door criterion.)

20.2 Prove Eq. 20.3, by using the causal Markov property of the appropriate surgically-altered graph.

1. The variable T contains all the parents of X; V contains all variables other than X, Y, and T. Explain why

$$\Pr\left(Y=y, X=x', T=t, V=v|do(X=x)\right) = \delta_{xx'} \frac{\Pr(Y=y, X=x, T=t, V=v)}{\Pr(X=x|T=t)} \tag{20.43}$$

where δ_{ij} is the "Kronecker delta", 1 when $i=j$ and 0 when $i \neq j$.
Hint: The left-hand side of the equation has to factor according to the graph we get after intervening on X, and the probability in the numerator on the right-hand side comes from the graphical model before the intervention. How do they differ?

[21] As I once heard a distinguished public health expert put it, "This problem is too important to worry about getting it right."

[22] "My boss/textbook says so" and "so I can estimate β" are not good reasons

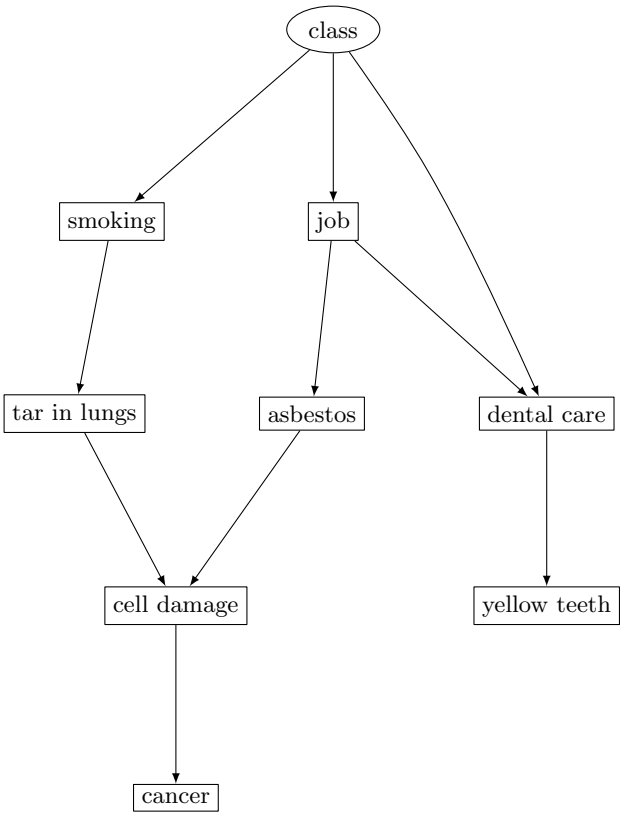


Figure 20.10 DAG for Exercise 20.3.

2. Assuming Eq. 20.43 holds, show that

$$\Pr\left(Y=y, X=x', T=t, V=v|do(X=x)\right) = \delta_{xx'}\Pr\left(Y=y, X=x, T=t, V=v|X=x, T=t\right)\Pr\left(T=t\right) \tag{20.44}$$

Hint: $\Pr\left(A|B\right) = \Pr\left(A, B\right)/\Pr\left(B\right)$.

3. Assuming Eq. 20.44 holds, use the law of total probability to derive Eq. 20.3, i.e., to derive

$$\Pr\left(Y=y|do(X=x)\right) = \sum_t \Pr\left(Y=y|X=x, T=t\right)\Pr\left(T=t\right) \tag{20.45}$$

20.3 Refer to Figure 20.10. Can we use the front door criterion to estimate the effect of occupational prestige on cancer? If so, give a set of variables which we would use as mediators. Is there more than one such set? If so, can you find them all? Are there variables we could add to this set (or sets) which would violate the front-door criterion?

20.4 Solve Eq. 20.27 for $\Pr\left(Y=1|do(X=0)\right)$ and $\Pr\left(Y=1|do(X=1)\right)$ in terms of the other conditional probabilities. When is the solution unique?

20.5 (*Lengthy, conceptual, open-ended*) Read Salmon (1984). When does his "statistical relevance basis" provide enough information to identify causal effects?

21

Estimating Causal Effects from Observations

Chapter 20 gave us ways of identifying causal effects, that is, of knowing when quantities like $\Pr(Y = y|do(X = x))$ are functions of the distribution of observable variables. Once we know that something is identifiable, the next question is how we can actually estimate it from data.

21.1 Estimators in the Back- and Front- Door Criteria

The back-door and front-door criteria for identification not only show us when causal effects are identifiable, they actually give us formulas for representing the causal effects in terms of ordinary conditional probabilities. When S satisfies the back-door criterion (Chapter 14), we can use parametric density models, we can model $Y|X, S = f(X, S) + \epsilon_Y$ and use regression, etc. If $\widehat{\Pr}(Y = y|X = x, S = s)$ is a consistent estimator of $\Pr(Y = y|X = x, S = s)$, and $\widehat{\Pr}(S = s)$ is a consistent estimator of $\Pr(S = s)$, then

$$\sum_s \widehat{\Pr}(S = s)\, \widehat{\Pr}(Y = y|X = x, S = s) \tag{21.1}$$

will be a consistent estimator of $\Pr(Y|do(X = x))$.

In principle, I could end this section right here, but there are some special cases and tricks which are worth knowing about. For simplicity, I will in this section only work with the back-door criterion, since estimating with the front-door criterion amounts to doing two rounds of back-door adjustment.

21.1.1 Estimating Average Causal Effects

Because $\Pr(Y|do(X = x))$ is a probability distribution, we can ask about $\mathbb{E}[Y|do(X = x)]$, when it makes sense for Y to have an expectation value; it's just

$$\mathbb{E}[Y|do(X = x)] = \sum_y y \Pr(Y = y|do(X = x)) \tag{21.2}$$

as you'd hope. This is the **average effect**, or sometimes just **the effect** of $do(X = x)$. While it is certainly not *always* the case that it summarizes all there is to know about the effect of X on Y, it is often useful.

If we identify the effect of X on Y through the back-door criterion, with control

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

variables S, then some algebra shows

$$\mathbb{E}[Y|do(X=x)] = \sum_y y \Pr(Y=y|do(X=x)) \tag{21.3}$$

$$= \sum_y y \sum_s \Pr(Y=y|X=x, S=s)\Pr(S=s) \tag{21.4}$$

$$= \sum_s \Pr(S=s) \sum_y y \Pr(Y=y|X=x, S=s) \tag{21.5}$$

$$= \sum_s \Pr(S=s)\mathbb{E}[Y|X=x, S=s] \tag{21.6}$$

The inner conditional expectation is just the regression function $\mu(x,s)$, for when we try to make a point-prediction of Y from X and S, so now all of the regression methods from Part I come into play. We would, however, still need to know the distribution $\Pr(S)$, so as to average appropriately. Let's turn to this.

21.1.2 Avoiding Estimating Marginal Distributions

We'll continue to focus on estimating the causal effect of X on Y using the back-door criterion, i.e., assuming we've found a set of control variables S such that

$$\Pr(Y=y|do(X=x)) = \sum_s \Pr(Y=y|X=x, S=s)\Pr(S=s) \tag{21.7}$$

S will generally contain multiple variables, so we are committed to estimating two potentially quite high-dimensional distributions, $\Pr(S)$ and $\Pr(Y|X,S)$. Even assuming that we knew all the distributions, just enumerating possible values s and summing over them would be computationally demanding. (Similarly, if S is continuous, we would need to do a high-dimensional integral.) Can we reduce these burdens?

One useful short-cut is to use the law of large numbers, rather than exhaustively enumerating all possible values of s. Notice that the left-hand side fixes y and x, so $\Pr(Y=y|X=x,S=s)$ is just some function of s. If we have an IID sample of realizations of S, say $s_1, s_2, \ldots s_n$, then the law of large numbers says that, for all well-behaved function f,

$$\frac{1}{n}\sum_{i=1}^{n} f(s_i) \to \sum_s f(s)\Pr(S=s) \tag{21.8}$$

Therefore, with a large sample,

$$\Pr(Y=y|do(X=x)) \approx \frac{1}{n}\sum_{i=1}^{n} \Pr(Y=y|X=x, S=s_i) \tag{21.9}$$

and this will still be (approximately) true when we use a consistent estimate of the conditional probability, rather than its true value.

The same reasoning applies for estimating $\mathbb{E}[Y|do(X=x)]$. Moreover, we can use the same reasoning to avoid explicitly summing over all possible s if we

do have $\Pr(S)$, by simulating from it[1]. Even if our sample (or simulation) is not completely IID, but is statistically stationary, in the sense we will cover in Chapter 23 (strictly speaking: "ergodic"), then we can still use this trick.

None of this gets us away from having to estimate $\Pr(Y|X,S)$, which is still going to be a high-dimensional object, if S has many variables.

21.1.3 Matching

Suppose that our causal variable of interest X is binary, or (almost equivalent) that we are only interested in comparing the effect of two levels, $do(X=1)$ and $do(X=0)$. Let's call these the "treatment" and "control" groups for definiteness, though nothing really hinges on one of them being in any sense a normal or default value (as "control" suggests) — for instance, we might want to know not just whether men get paid more than women, but whether they are paid more *because* of their sex[2]. In situations like this, we are often not so interested in the full distributions $\Pr(Y|do(X=1))$ and $\Pr(Y|do(X=0))$, but just in the expectations, $\mathbb{E}[Y|do(X=1)]$ and $\mathbb{E}[Y|do(X=0)]$. In fact, we are often interested just in the *difference* between these expectations, $\mathbb{E}[Y|do(X=1)] - \mathbb{E}[Y|do(X=0)]$, what is often called the **average treatment effect**, or ATE.

Suppose we are the happy possessors of a set of control variables S which satisfy the back-door criterion. How might we use them to estimate this average treatment effect?

$$\begin{aligned} ATE &= \sum_s \Pr(S=s)\,\mathbb{E}[Y|X=1,S=s] - \sum_s \Pr(S=s)\,\mathbb{E}[Y|X=0,S=s] \quad (21.10) \\ &= \sum_s \Pr(S=s)\left(\mathbb{E}[Y|X=1,S=s] - \mathbb{E}[Y|X=0,S=s]\right) \quad (21.11) \end{aligned}$$

[1] This is a "Monte Carlo" approximation to the full expectation value.

[2] The example is both imperfect and controversial. It is imperfect because biological sex (never mind socio-cultural gender) is not *quite* binary, even in mammals, though the exceptional cases are quite rare. (See Dreger 1998 for a historical perspective.) It is controversial because many statisticians insist that there is no sense in talking about causal effects unless there is some actual manipulation or intervention one could do to change X for an actually-existing "unit" — see, for instance, Holland (1986), which seems to be the source of the slogan "No causation without manipulation". I will just note that (i) this is the kind of metaphysical argument which statisticians usually avoid (if we can't talk about sex or race as causes, because changing those makes the subject a "different person", how about native language? the shape of the nose? hair color? whether they go to college? age at which they started school? grades in school?); (ii) genetic variables are highly manipulable with modern experimental techniques, though we don't use those techniques on people; (iii) real scientists routinely talk about causal effects with no feasible manipulation (e.g., "continental drift causes earthquakes"), or even imaginable manipulation (e.g., "the solar system formed because of gravitational attraction"). It may be merely coincidence that (iv) many of the statisticians who make such pronouncements work or have worked for the Educational Testing Service, an organization with an interest in asserting that, strictly speaking, sex and race cannot have any *causal* role in the score anyone gets on the SAT. (Points (i)–(iii) follow Glymour (1986); Glymour and Glymour (2014); Marcellesi (2013).)

Abbreviate $\mathbb{E}[Y|X=x, S=s]$ as $\mu(x,s)$, so that the average treatment effect is

$$\sum_s (\mu(1,s) - \mu(0,s))\Pr(S=s) = \mathbb{E}[\mu(1,S) - \mu(0,S)] \tag{21.12}$$

Suppose we got to observe μ. Then we could use the law of large numbers argument above to say

$$ATE \approx \frac{1}{n}\sum_{i=1}^{n} \mu(1,s_i) - \mu(0,s_i) \tag{21.13}$$

Of course, we don't get to see either $\mu(1,s_i)$ or $\mu(0,s_i)$. We don't even get to see $\mu(x_i,s_i)$. At best, we get to see $Y_i = \mu(x_i,s_i) + \epsilon_i$, with ϵ_i being mean-zero noise.

Clearly, we need to estimate $\mu(1,s_i) - \mu(0,s_i)$. In principle, any consistent estimator of the regression function, $\widehat{\mu}$, would do. If, for some reason, you were scared of doing a regression, however, the following scheme might occur to you: First, find all the units in the sample with $S=s$, and compare the mean Y for those who are treated ($X=1$) to the mean Y for those who are untreated ($X=0$). Writing the the set of units with $X=1$ and $S=s$ as $\mathcal{T}_s$, and the set of units with $X=0$ and $S=s$ as $\mathcal{C}_s$, then

$$\sum_s \left(\frac{1}{|\mathcal{T}_s|}\sum_{i\in\mathcal{T}_s} Y_i - \frac{1}{|\mathcal{C}_s|}\sum_{j\in\mathcal{C}_s} Y_j\right)\Pr(S=s) \tag{21.14}$$

$$= \sum_s \left(\frac{1}{|\mathcal{T}_s|}\sum_{i\in\mathcal{T}_s} \mu(1,s)+\epsilon_i - \frac{1}{|\mathcal{C}_s|}\sum_{j\in\mathcal{C}_s} \mu(0,s)+\epsilon_j\right)\Pr(S=s) \tag{21.15}$$

$$= \sum_s (\mu(1,s)-\mu(0,s))\Pr(S=s) + \sum_s \left(\frac{1}{|\mathcal{T}_s|}\sum_{i\in\mathcal{T}_s}\epsilon_i - \frac{1}{|\mathcal{C}_s|}\sum_{j\in\mathcal{C}_s}\epsilon_j\right)\Pr(S=s) \tag{21.16}$$

The first part is what we want, and the second part is an average of noise terms, so it goes to zero as $n\to\infty$. Thus we have a consistent estimator of the average treatment effect.

We could however go further. Take any unit i where $X=1$; it has some value s_i for the covariates. Suppose we can find another unit i^* with the same value of the covariates, but with $S=0$. Then

$$Y_i - Y_{i^*} = \mu(1,s_i) + \epsilon_i - \mu(0,s_i) - \epsilon_{i^*} \tag{21.17}$$

The comparison between the response of the treated unit and this **matched** control unit is an unbiased estimate of $\mu(1,s_i)-\mu(0,s_i)$. If we can find a match i^* for every unit i, then

$$\frac{1}{n}\sum_{i=1}^{n} Y_i - Y_{i^*} \tag{21.18}$$

$$= \frac{1}{n}\sum i = 1^n \mu(1,s_i) - \mu(0,s_i) + \frac{1}{n}\sum_{i=1}^{n}\epsilon_i \tag{21.19}$$

The first average is, by the law-of-large-numbers argument, approximately the

average treatment effect, and the second is the average of noise terms, so it should be going to zero as $n \to \infty$. Thus, matching gives us a consistent estimate of the average treatment effect, without any *explicit* regression. Instead, we rely on a paired comparison, because members of the treatment group are being compared to with members of the control group with matching values of the covariates S. This often works vastly better than estimating μ through a linear model.

There are three directions to go from here. One is to deal with all of the technical problems and variations which can arise. We might match each unit against multiple other units, to get further noise reduction. If we can't find an exact match, the usual approach is to match each treated unit against the control-group unit with the closest values of the covariates. Exploring these details is important to applications, but we won't follow it up here (see further readings).

A second direction is to remember that matching does not solve the identification problem. Computing Eq. 21.19 only gives us an estimate of the average treatment effect if S satisfies the back-door criterion. If S does not, then even if matching is done perfectly, Eq. 21.19 does nothing of any particular interest. Matching is one way of estimating *identified* average treatment effects; it contributes nothing to *solving* identification problems.

Third, and finally, matching is really doing nearest neighbor regression (§1.5.1). To get the difference between the responses of treated and controlled units, we're comparing each treated unit to the control-group unit with the closest values of the covariates. When people talk about *matching* estimates of average treatment effects, they usually mean that the number of nearest neighbors we use for each treated unit is fixed as n grows.

Once we realize that matching is really just nearest-neighbor regression, it may become less compelling; at the very least many issues should come to mind. As we saw in §1.5.1, to get consistent estimates of μ out of k-nearest neighbors, we need to let k grow (slowly) with n. If k is fixed, then the bias of $\widehat{\mu}(x, s)$ is either zero or goes quickly to zero as n grows (quicker the smaller k is), but $\mathbb{V}\left[\widehat{\mu}x, s\right] \not\to 0$ as $n \to \infty$. If all we want to do is estimate the average treatment effect, this remaining asymptotic variance at each s will still average out, but it would be a problem if we wanted to look at anything more detailed. More generally, the bias-variance tradeoff is a *tradeoff*, and it's not always a good idea to prioritize low bias over anything else. Moreover, it's not exactly clear that we *should* use a fixed k, or for that matter should use nearest neighbors instead of any other consistent regression method.

Nearest neighbor regression, like every other nonparametric method, is subject to the curse of dimensionality[3]; therefore, so is matching[4]. It would be very nice

[3] An important caveat: when S is high-dimensional but all the data fall on or very near a low-dimensional sub-space, nearest neighbor regression will adapt to this low effective dimensionality (Kpotufe, 2011). Not all regression methods have this nice property.

[4] If we can could do matching easily for high-dimensional S, then we could match treated units to other treated units, and control-group units to control-group units, and do easy high-dimensional regression. Since we know high-dimensional regression is hard, and we just reduced regression to matching, high-dimensional matching must be at least as hard.

if there was some way of lightening the curse when estimating treatment effects. We'll turn to that next.

21.1.4 Propensity Scores

The problems of having to estimate high-dimensional conditional distributions and of averaging over large sets of control values are both reduced if the set of control variables has in fact only a few dimensions. If we have two sets of control variables, S and R, both of which satisfy the back-door criterion for identifying $\Pr(Y|do(X=x))$, all else being equal we should use R if it contains fewer variables than S[5]

An important special instance of this is when we can set $R = f(S)$, for some function S, and have

$$X \perp\!\!\!\perp S|R \tag{21.20}$$

In the jargon, R is a **sufficient statistic**[6] for predicting X from S. To see why this matters, suppose now that we try to identify $\Pr(Y=y|do(X=x))$ from a back-door adjustment for R alone, not for S. We have[7]

$$\sum_r \Pr(Y|X=x, R=r)\Pr(R=r) \tag{21.21}$$

$$= \sum_{r,s} \Pr(Y, S=s|X=x, R=r)\Pr(R=r)$$

$$= \sum_{r,s} \Pr(Y|X=x, R=r, S=s)\Pr(S=s|X=x, R=r)\Pr(R=r) \tag{21.22}$$

$$= \sum_{r,s} \Pr(Y|X=x, S=s)\Pr(S=s|X=x, R=r)\Pr(R=r) \tag{21.23}$$

$$= \sum_{r,s} \Pr(Y|X=x, S=s)\Pr(S=s|R=r)\Pr(R=r) \tag{21.24}$$

$$= \sum_s \Pr(Y|X=x, S=s)\sum_r \Pr(S=s, R=r) \tag{21.25}$$

$$= \sum_s \Pr(Y|X=x, S=s)\Pr(S=s) \tag{21.26}$$

$$= \Pr(Y|do(X=x)) \tag{21.27}$$

That is to say, if S satisfies the back-door criterion, then so does R. Since R is a function of S, both the computational and the statistical problems which come from using R are no worse than those of using S, and possibly much better, if R has much lower dimension.

[5] Other things which might not be equal: the completeness of data on R and S; parametric assumptions might be more plausible for the variables in S, giving a better rate of convergence; we might be more confident that S really does satisfy the back-door criterion.

[6] This is not the same sense of the word "sufficient" as in "causal sufficiency".

[7] Going from Eq. 21.22 to Eq. 21.23 uses the fact that $R = f(S)$, so conditioning on both R and S is the same as just conditioning on S. Going from Eq. 21.23 uses the fact that $S \perp\!\!\!\perp X|R$.

It may seem far-fetched that such a summary score should exist, but really all that's required is that some combinations of the variables in S carry the same information about X as the whole of S does. Consider for instance, the set-up where

$$X \leftarrow \sum_{j=1}^{p} V_j + \epsilon_X \tag{21.28}$$

$$Y \leftarrow f(X, V_1, V_2, \ldots V_p) + \epsilon_Y \tag{21.29}$$

To identify the effect of X on Y, we need to block the back-door paths between them. Each one of the V_j provides such a back-door path, so we need to condition on *all* of them. However, if $R = \sum_{j=1}^{p} V_j$, then $X \perp\!\!\!\perp \{V_1, V_2, \ldots V_p\} | R$, so we could reduce a p-dimensional set of control variables to a one-dimensional set.

Often, as here, finding summary scores will depend on the functional form, and so not be available in the general, non-parametric case. There is, however, an important special case where, if we can use the back-door criterion at all, we can use a one-dimensional summary.

This is the case where X is binary. If we set $f(S) = \Pr(X = 1|S = s)$, and then take this as our summary R, it is not hard to convince oneself that $X \perp\!\!\!\perp S|R$ (Exercise 21.1). This $f(S)$ is called the **propensity score**. It is remarkable, and remarkably convenient, that an arbitrarily large set of control variables S, perhaps with very complicated relationships with X and Y, can always be boiled down to a single number between 0 and 1, but there it is.

That said, except in very special circumstances, there is no analytical formula for $f(S)$. This means that it must be modeled and estimated. The most common model used is logistic regression, but so far as I can see this is just because many people know no other way to model a binary outcome. Since accurate propensity scores are needed to make the method work, it would seem to be worthwhile to model R very carefully, and to consider GAM or fully non-parametric estimates. If S contains a lot of variables, then estimating $\Pr(X = 1|S = s)$ is a high-dimensional regression problem, and so itself subject to the curse of dimensionality.

21.1.5 Propensity Score Matching

If the number of covariates in S is large, the curse of dimensionality settles upon us. Many values of S will have few or no individuals at all in the data set, let alone a large number in both the treatment and the control groups. Even if the real difference $\mathbb{E}[Y|X = 1, S = s] - \mathbb{E}[Y|X = 0, S = s]$ is small, with only a few individuals in either sub-group we could easily get a large difference in sample means. And of course with continuous covariates in S, each individual will generally have no exact matches at all.

The very clever idea of Rosenbaum and Rubin (1983) is to ameliorate this by matching not on S, but on the propensity score $R = \Pr(X = 1|S)$ defined above (p. 491). We have seen already that when X is binary, adjusting for the

propensity score is just as good as adjusting for the full set of covariates S. It is easy to double-check (Exercise 21.2) that

$$\begin{aligned}&\sum_s \Pr(S=s)\left(\mathbb{E}[Y|X=1,S=s]-\mathbb{E}[Y|X=0,S=s]\right)\\&=\sum_r \Pr(R=r)\left(\mathbb{E}[Y|X=1,R=r]-\mathbb{E}[Y|X=0,R=r]\right)\end{aligned} \quad (21.30)$$

when $R = \Pr(X=1|S=s)$, so we lose nothing, for these purposes, by matching on the propensity score R rather than on the covariates S. Intuitively, we now compare each treated individual with one who was just as likely to have received the treatment, but, by chance, did not[8]. On average, the differences between such matched individuals have to be due to the treatment.

What have we gained by doing this? Since R is always a one-dimensional variable, no matter how big S is, it is going to be *much* easier to find matches on R than on S. This does not actually break the curse of dimensionality, but rather shifts its focus, from the regression of Y on X and S to the regression of X on S. Still, this can be a very real advantage.

It is important to be clear, however, that the gain here is in computational tractability and (perhaps) statistical efficiency, not in fundamental identification. With $R = \Pr(X=1|S=s)$, it will always be true that $X \perp\!\!\!\perp S|R$, *whether or not* the back-door criterion is satisfied. If the criterion is satisfied, in principle there is nothing stopping us from using matching on S to estimate the effect, except our own impatience. If the criterion is not satisfied, having a compact one-dimensional summary of the wrong set of control variables is just going to let us get the wrong answer faster.

Some confusion seems to have arisen on this point, because, conditional on the propensity score, the treated group and the control group have the same distribution of covariates. (Again, recall that $X \perp\!\!\!\perp S|R$.) Since treatment and control groups have the same distribution of covariates in a randomized experiment, some people have concluded that propensity score matching is just as good as randomization[9]. This is emphatically *not* the case.

21.2 Instrumental-Variables Estimates

§20.3.3 introduced the idea of using instrumental variables to identify causal effects. Roughly speaking, I is an instrument for identifying the effect of X on Y when I is a cause of X, but the only way I is associated with Y is through directed paths which go through X. To the extent that variation in I predicts variation in X and Y, this can only be because X has a causal influence on Y. More precisely, given some controls S, I is a valid instrument when $I \not\perp\!\!\!\perp X|S$, and every path from I to Y left open by S has an arrow into X.

[8] Methods of approximate matching often work better on propensity scores than on the full set of covariates, because the former are lower-dimensional.

[9] These people do not include Rubin and Rosenbaum, but it is easy to see how their readers could come away with this impression. See Pearl (2009b, §11.3.5), and especially Pearl (2009a).

In the simplest case, of Figure 20.7, we saw that when everything is linear, we can find the causal coefficient of Y on X as

$$\beta = \frac{\text{Cov}\left[I, Y\right]}{\text{Cov}\left[I, X\right]} \tag{21.31}$$

A one-unit change in I causes (on average) an α-unit change in X, and an $\alpha\beta$-unit change in Y, so β is, as it were, the gearing ratio or leverage of the mechanism connecting I to Y.

Estimating β by plugging in the sample values of the covariances into Eq. 21.31 is called the **Wald estimator** of β. In more complex situations, we might have multiple instruments, and be interested in the causal effects of multiple variables, and we might have to control for some covariates to block undesired paths and get valid instruments. In such situations, the Wald estimator breaks down.

There is however a more general procedure which still works, provided the linearity assumption holds. This is called **two-stage regression**, or **two-stage least squares** (2SLS).

1. Regress X on I and S. Call the fitted values $\hat{x}$.
2. Regress Y on $\hat{x}$ and S, but *not* on I. The coefficient of Y on $\hat{x}$ is a consistent estimate of β.

The logic is very much as in the Wald estimator: conditional on S, variations in I are independent of the rest of the system. The only way they can affect Y is through their effect on X. In the first stage, then, we see how much changes in the instruments affect X. In the second stage, we see how much these I-caused changes in X change Y; and this gives us what we want.

To actually prove that this works, we would need to go through some heroic linear algebra to show that the population version of the two-stage estimator is actually equal to β, and then a straight-forward argument that plugging in the appropriate sample covariance matrices is consistent. The details can be found in any econometrics textbook, so I'll skip them. (But see Exercise 21.4.)

As mentioned in §21.2, there are circumstances where it is possible to use instrumental variables in nonlinear and even nonparametric models. The technique becomes far more complicated, however, because finding $\Pr\left(Y = y|do(X = x)\right)$ requires solving Eq. 20.15,

$$\Pr\left(Y|do(I = i)\right) = \sum_x \Pr\left(Y|do(X = x)\right) \Pr\left(X = x|do(I = i)\right)$$

and likewise finding $\mathbb{E}\left[Y|do(X = x)\right]$ means solving

$$\mathbb{E}\left[Y|do(I = i)\right] = \sum_x \mathbb{E}\left[Y|do(X = x)\right] \Pr\left(X = x|do(I = i)\right) \tag{21.32}$$

When, as is generally the case, x is continuous, we have rather an integral equation,

$$\mathbb{E}\left[Y|do(I = i)\right] = \int \mathbb{E}\left[Y|do(X = x)\right] p(x|do(I = i))dx \tag{21.33}$$

Solving such integral equations is not (in general) impossible, but it is hard, and the techniques needed are much more complicated than even two-stage least squares. I will not go over them here, but see Li and Racine (2007, chs. 16–17).

21.3 Uncertainty and Inference

The point of the identification strategies from Chapter 20 is to reduce the problem of causal inference to that of ordinary statistical inference. Having done so, we can assess our uncertainty about any of our estimates of causal effects the same way we would assess any other statistical inference. If we want confidence intervals or standard errors for $\mathbb{E}[Y|do(X=1)] - \mathbb{E}[Y|do(X=0)]$, for instance, we can treat our estimate of this like any other point estimate, and proceed accordingly. In particular, we can use the bootstrap (Chapter 6), if analytical formulas are unavailable or unappealing.

The one wrinkle to the use of analytical formulas comes from two-stage least-squares. Taking standard errors, confidence intervals, etc., for β from the usual formulas for the second regression neglects the fact that this estimate of β comes from regressing Y on $\hat{x}$, which is itself an estimate and so uncertain. Even if this is handled with some care, two-stage least squares is extraordinarily vulnerable to any violations in the usual assumptions about IID Gaussian errors. Young (2017), reviewing over 1000 (!) instrumental-variable regressions from top economics journals, shows that this is not merely a theoretical concern, but undermines a huge amount of the published literature.

21.4 Recommendations

Instrumental variables are a very clever idea, but they need to be treated with caution. They only work if the instruments are valid, and that validity rests just on assumptions about the causal structure. The crucial point, after all, is that the instrument is an indirect cause of Y, but *only* through X, with no other (unblocked) paths connecting I to Y. This can only too easily fail, if some indirect path has been neglected. They also require great care in their statistical inference (Young, 2017).

Matching, especially propensity score matching, is just as ingenious, and just as much at the mercy of the correctness of the DAG. Whether we match directly on covariates, or indirectly through the propensity score, what matters is whether the covariates really block off the back-door pathways between X and Y. If the covariates block those pathways, well and good; any consistent form of regression will work, including one called "matching" because "nonparametric nearest-neighbor smoothing" sounds too scary. If the covariates do not block the back-door pathways, then no amount of statistical ingenuity is going to help you.

There is a curious divide, among practitioners, between those who lean mostly on instrumental variables, and those who lean mostly on matching. The former

tend to suspect that (in our terms) the covariates used in matching are not enough to block all the back-door paths[10], *and* to think that the work is more or less over once an exogenous variable has been found. The matchers, for their part, think the instrumentalists are too quick to discount the possibility that their instruments are connected to Y through unmeasured pathways[11], but that if you match on enough variables, you've got to block the back-door paths. (They don't often worry that in doing so they might be activating colliders, or blocking front-door paths.) As is often the case in science, there is much truth to each faction's criticism of the other side. *You* are now in a position to think more clearly about these matters, and to act more intelligently, than many practitioners.

Throughout these chapters, we have been assuming that we know the correct DAG. Without such assumptions, or ones equivalent to them, none of these ideas can be used. In the next chapter, then, we will look at how to actually begin *discovering* causal structure from data.

21.5 Further Reading

The material in §21.1 is largely "folklore", though see Morgan and Winship (2007), which also treats instrumental variable estimation, and a number of other, more specialized techniques, like "regression discontinuity designs" and "difference in differences". It does not, however, consider nonparametric regression methods.

On matching, Stuart (2010) is another good review, including software as well as methods. For some of the asymptotic theory, including the connection to nearest neighbor methods, see Abadie and Imbens (2006).

The propensity score matching method has become incredibly popular since

[10] As an example for their side, Arceneaux *et al.* (2010) applied matching methods to an actual experiment, where the real causal relations could be worked out straightforwardly. Well-conduced propensity-score "matching suggests that [a] pre-election phone call that encouraged people to wear their seat belts also generated huge increases in voter turnout". The paper gives a convincing explanation of where this illusory effect comes from, i.e., of what the unblocked back-door path is, which I will not spoil for you.

[11] For instance, a widely-promoted preprint by three economists argued that watching television caused autism in children. (I leave tracking down the manuscript as an exercise for the reader.) The economists used the variation in how much it rains across different locations in the states of California, Oregon and Washington as an instrument (I) to predict average TV-watching (X) and its affects on the prevalence of autism (Y). It is certainly plausible that kids watch more TV when it rains, and that neither TV-watching nor autism causes rain. But this leaves open the question of whether rain and the prevalence of autism might not have some common cause, and for the west coast of the US in particular it is easy to find one. It is well-established that the risk of autism is higher among children of older parents, and that more-educated people tend to have children later in life. All three states have, of course, a striking contrast between large, rainy cities full of educated people (San Francisco, Portland, Seattle), and very dry, very rural locations on the other side of the mountains. Thus there is a (potential) uncontrolled common cause of rain and autism, namely geographic location, and the situation is as in Figure 20.8. — For a rather more convincing effort to apply ideas about causal inference to understanding the changing prevalence of autism, see Liu *et al.* (2010).

Rosenbaum and Rubin (1983), and there are a huge number of implementations of various versions of it. The `optmatch` package in R is notable for doing the actual matching in an extremely flexible and efficient way, but leaves defining matching criteria largely to the user (Hansen and Klopfer, 2006). The `MatchIt` package (Ho *et al.*, 2011) includes more tools for actually calculating propensity scores or other mesures of similarity, and then doing the matching. Stuart (2010) is also good on relevant software in R and other languages.

Rubin and Waterman (2006) is an extremely clear and easy-to-follow introduction to propensity score matching as a method of causal inference; Imbens and Rubin (2015) is a more comprehensive presentation of the work done by Rubin, Imbens and collaborators on estimating causal effects by matching, propensity scores, and instrumental variables. (Many of the original papers are reprinted in Rubin 2006.) While sound on theory, that book's worked examples cannot be recommended as examples of statistical craft (Shalizi, 2016).

King and Nielsen (2016) is an interesting argument against matching on propensity scores, in favor of matching on the full set of covariates, related to the extra variance of estimating the propensity scores.

Exercises

21.1 Suppose X is binary, and define $R = \Pr(X = 1|S)$. Show that $X \perp\!\!\!\perp S|R$. .

21.2 Prove Eq. 21.30.

21.3 Suppose that X has three levels, say $0, 1, 2$. Let R be the vector $(\Pr(X = 0|S = s), \Pr(X = 1|S = s))$. Prove that $X \perp\!\!\!\perp S|R$. (This is how to generalize propensity scores to non-binary X.)

21.4 For the situation in Figure 20.7, prove that the two-stage least-squares estimate of β is the same as the Wald estimate.

22

Discovering Causal Structure from Observations

[[ATTN: Further examples]]

The last few chapters have, hopefully, convinced you that when you want to do causal inference, it would help to know the causal graph. We have seen how the graph would let us calculate the effects of actual or hypothetical manipulations of the variables in the system. Furthermore, the graph tells us about what effects we can and cannot identify, and estimate, from observational data. But everything has posited that we know the graph somehow. This chapter finally deals with where the graph comes from.

There are fundamentally three ways to get the DAG:

- Prior knowledge
- Guessing-and-testing
- Discovery algorithms

Prior knowledge

There's little to say, here, about the first, because, while it's important, it's not very statistical. As functioning adult human beings, you have a lot of everyday causal knowledge, which doesn't disappear the moment you start doing data analysis. Moreover, you are the inheritor of a vast scientific tradition which has, through patient observation, toilsome experiments, ingenious theorizing and intricate debate, acquired even more causal knowledge. You can and should use this. Someone's sex or race or caste at birth might be causes of the job they get or their income at age 30, but not the other way around. Running an electric current through a wire produces heat at a rate proportional to the square of the magnitude of current. Malaria is due to a parasite transmitted by mosquitoes, and spraying mosquitoes with insecticides makes the survivors more resistant to those chemicals. All of these sorts of ideas can be expressed graphically, or at least as constraints on graphs.

We can, and should, also use graphs to represent scientific ideas which are not as secure as Joule's law or the epidemiology of malaria. The ideas people work with in areas like psychology or economics, are really quite tentative, but they are ideas about the causal structure of parts of the world, and so graphical models are implicit in them.

All of which said, even if we *think* we know very well what's going on, we will often still want to check it, and that brings us the guess-and-test route.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

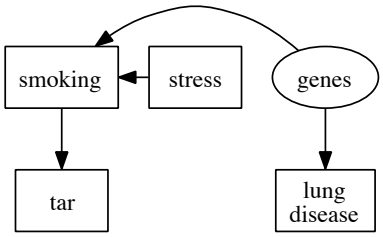


Figure 22.1 A hypothetical causal model in which smoking is associated with lung disease, but does not cause it. Rather, both smoking and lung disease are caused by common genetic variants. (This idea was due to R. A. Fisher.) Smoking is also caused, in this model, by stress.

22.1 Testing DAGs

A graphical causal model makes two kinds of qualitative claims. One is about direct causation. If the model says X is a parent of Y, then it says that changing X will change the (distribution of) Y. If we experiment on X (alone), moving it back and forth, and yet Y is unaltered, we know the model is wrong and can throw it out.

The other kind of claim a DAG model makes is about probabilistic conditional independence. If S d-separates X from Y, then $X \perp\!\!\!\perp Y|S$. If we observed X, Y and S, and see that $X \not\perp\!\!\!\perp Y|S$, then we know the model is wrong and can throw it out. (More: we know that there is a path linking X and Y which isn't blocked by S.) Thus in the model of Figure 22.1, lungdisease $\perp\!\!\!\perp$ tar|smoking. If lung disease and tar turn out to be dependent when conditioning on smoking, the model must be wrong.

This then is the basis for the guess-and-test approach to getting the DAG:

- Start with an initial guess[1] about the DAG.
- Deduce conditional independence relations from d-separation.
- Test these, and reject the DAG if variables which ought to be conditionally independent turn out to be dependent.

This is a distillation of primary-school scientific method: formulate a hypotheses (the DAG), work out what the hypothesis implies, test those predictions, reject hypotheses which make wrong predictions.

It may happen that there are only a few competing, scientifically-plausible models, and so only a few, competing DAGs. Then it is usually a good idea to focus on checking predictions which *differ* between them. So in both Figure 22.1

[1] We'll come back to where this guess might come from. One possibility, if you're really stumped, is just to start enumerating all the possible DAGs on the variables you are concerned with. But this grows *very* rapidly with the number of variables, and I don't *really* recommend it.

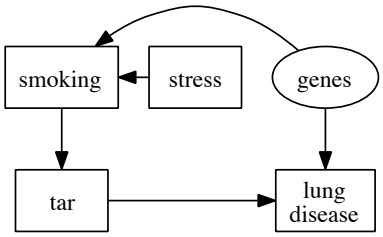


Figure 22.2 As in Figure 22.1, but now tar in the lungs does cause lung disease.

and in Figure 22.2, stress $\perp\!\!\!\perp$ tar|smoking. Checking that independence thus does nothing to help us distinguish between the two graphs. In particular, confirming that stress and tar are independent given smoking really doesn't give us evidence *for* the model from Figure 22.1, since it equally follows from the other model. If we want such evidence, we have to look for something they *disagree* about.

In any case, testing a DAG means testing conditional independence, so let's turn to that next.

22.2 Testing Conditional Independence

Recall from §18.4 that conditional independence is equivalent to zero conditional mutual information: $X \perp\!\!\!\perp Y|Z$ if and only if $I[X;Y|Z] = 0$. In principle, this solves the problem. In practice, estimating mutual information is non-trivial, and in particular the sample mutual information often has a very complicated distribution. You *could* always bootstrap it, but often something more tractable is desirable. Completely general conditional independence testing is actually an active area of research. Some of this work is still quite mathematical (Sriperumbudur *et al.*, 2010), but it has already led to practical tests (Székely and Rizzo, 2009; Gretton *et al.*, 2012; Zhang *et al.*, 2011) and no doubt more are coming soon.

If all your variables are discrete, you just have a big contingency table problem, and could use a G^2 or χ^2 test. If everything is linear and multivariate Gaussian, $X \perp\!\!\!\perp Y|Z$ is equivalent to zero partial correlation[2]. Nonlinearly, if $X \perp\!\!\!\perp Y|Z$, then $\mathbb{E}[Y \mid Z] = \mathbb{E}[Y \mid X, Z]$, so if smoothing Y on X and Z leads to different predictions than just smoothing on Z, conditional independence fails. To reverse this, and go from $\mathbb{E}[Y \mid Z] = \mathbb{E}[Y \mid X, Z]$ to $X \perp\!\!\!\perp Y|Z$, requires the extra assumption that Y doesn't depend on X through its variance or any other moment. (This is weaker than the linear-and-Gaussian assumption, of course.)

[2] Recall that the partial correlation between X and Y given Z is the correlation between X and Y, after linearly regressing each of them on Z separately. That is, it is the correlation of their residuals.

The conditional independence relation $X \perp\!\!\!\perp Y|Z$ is fully equivalent to $\Pr(Y \mid X, Z) = \Pr(Y \mid Z)$. We could check this using non-parametric density estimation, though we would have to bootstrap the distribution of the test statistic. A more automatic, if slightly less rigorous, procedure comes from the idea mentioned in §14.5: If X is in fact useless for predicting Y given Z, then an adaptive bandwidth selection procedure (like cross-validation) should realize that giving any finite bandwidth to X just leads to over-fitting. The bandwidth given to X should tend to the maximum allowed, smoothing X away altogether. This argument can be made more formal, and made into the basis of a test (Hall *et al.*, 2004; Li and Racine, 2007).

22.3 Faithfulness and Equivalence

In graphical models, d-separation implies conditional independence: if S blocks all paths from U to V, then $U \perp\!\!\!\perp V|S$. To reverse this, and conclude that if $U \perp\!\!\!\perp V|S$ then S must d-separate U and V, we need an additional assumption, already referred to in §19.2, called **faithfulness**. More exactly, if the distribution is faithful to the graph, then if S does not d-separate U from V, $U \not\perp\!\!\!\perp V|S$. The combination of faithfulness and the Markov property means that $U \perp\!\!\!\perp V|S$ if and only if S d-separates U and V.

This seems extremely promising. We can test whether $U \perp\!\!\!\perp V|S$ for any sets of variables we like. We could in particular test whether each pair of variables is independent, given all sorts of conditioning variable sets S. If we assume faithfulness, when we find that $X \perp\!\!\!\perp Y|S$, we know that S blocks all paths linking X and Y, so we learn something about the graph. If $X \not\perp\!\!\!\perp Y|S$ for all S, we would seem to have little choice but to conclude that X and Y are directly connected. Might it not be possible to reconstruct or discover the right DAG from knowing all the conditional independence and dependence relations?

This is on the right track, but too hasty. Start with just two variables:

$$X \rightarrow Y \Rightarrow X \not\perp\!\!\!\perp Y \tag{22.1}$$

$$X \leftarrow Y \Rightarrow X \not\perp\!\!\!\perp Y \tag{22.2}$$

With only two variables, there is only one independence (or dependence) relation to worry about, and it's the same no matter which way the arrow points.

Similarly, consider these arrangements of three variables:

$$X \rightarrow Y \rightarrow Z \tag{22.3}$$

$$X \leftarrow Y \leftarrow Z \tag{22.4}$$

$$X \leftarrow Y \rightarrow Z \tag{22.5}$$

$$X \rightarrow Y \leftarrow Z \tag{22.6}$$

The first two are chains, the third is a fork, the last is a collider. It is not hard to check (Exercise 22.1) that the first three DAGs all imply exactly the same set

of conditional independence relations, which are different from those implied by the fourth[3]

These examples illustrate a general problem. There may be multiple graphs which imply the same independence relations, even when we assume faithfulness. When this happens, the exact same distribution of observables can factor according to, and be faithful to, all of those graphs. The graphs are thus said to be **equivalent**, or **Markov equivalent**. Observations alone cannot distinguish between equivalent DAGs. Experiment can, of course — changing Y alters both X and Z in a fork, but not a chain — which shows that there really is a difference between the DAGs, just not one *observational* data can track.

22.3.1 Partial Identification of Effects

Chapters 20–21 considered the identification and estimation of causal effects under the assumption that there was a single known graph. If there are multiple equivalent DAGs, then, as mentioned above, no amount of purely observational data can select a single graph. Background knowledge lets us rule out some equivalent DAGs[4], but it may not narrow the set of possibilities to a single graph. How then are we to actually do our causal estimation?

We *could* just pick one of the equivalent graphs, and do all of our calculations as though it were the only possible graph. This is often what people seem to do. The kindest thing one can say about it is that it shows confidence; phrases like "lying by omission" also come to mind.

A more principled alternative is to admit that the uncertainty about the DAG means that causal effects are only *partially* identified. Simply put, one does the estimation in each of the equivalent graphs, and reports the range of results[5]. If each estimate is consistent, then this gives a consistent estimate of the range of possible effects. Because the effects are not fully identified, this range will not narrow to a single point, even in the limit of infinite data, but admitting this, rather than claiming a non-existent precision, is simple scientific honesty.

22.4 Causal Discovery with Known Variables

Section 22.1 talks about how we can test a DAG, once we have it. This lets us eliminate some DAGs, but still leaves mysterious where they come from in the

[3] In all of the first three, $X \not\perp\!\!\!\perp Z$ but $X \perp\!\!\!\perp Z|Y$, while in the collider, $X \perp\!\!\!\perp Z$ but $X \not\perp\!\!\!\perp Z|Y$. Remarkably enough, the work which introduced the notion of forks and colliders, Reichenbach (1956), missed this — he thought that $X \perp\!\!\!\perp Z|Y$ in a collider as well as a fork. Arguably, this highly uncharacteristic mistake by a great scholar delayed the development of causal inference by thirty years or more, and is one of the reasons why, as Dean Eckles once put it, formal causal inference is an "idea behind its time" (`http://www.deaneckles.com/blog/429_ideas-behind-their-time-formal-causal-inference/`).

[4] If we know that X, Y and Z have to be in either a chain or a fork, with Y in the middle, and we know that X comes before Y in time, then we can rule out the fork and the chain $X \leftarrow Y \rightarrow Z$.

[5] Sometimes the different graphs will gave the same estimates of certain effects. For example, the chain $X \rightarrow Y \rightarrow Z$ and the fork $X \leftarrow Y \rightarrow Z$ will agree on the effect of Y on Z.

first place. While in principle there is nothing wrong which deriving your DAG from a vision of serpents biting each others' tails, so long as you test it, it would be nice to have a systematic way of finding good models. This is the problem of model discovery, and especially of causal discovery.

Causal discovery is silly with just one variable, and too hard for us with just two.[6]

With three or more variables, we have however a very basic principle. If there is no edge between X and Y, in either direction, then X is neither Y's parent nor its child. But any variable is independent of its non-descendants given its parents. Thus, for some set[7] of variables S, $X \perp\!\!\!\perp Y|S$ (Exercise 22.2). If we assume faithfulness, then the converse holds: if $X \perp\!\!\!\perp Y|S$, then there cannot be an edge between X and Y. Thus, there is no edge between X and Y if and only if we can make X and Y independent by conditioning on some S. Said another way, there is an edge between X and Y if and only if we cannot make the dependence between them go away, no matter what we condition on[8].

So let's start with three variables, X, Y and Z. By testing for independence and conditional independence, we could learn that there had to be edges between X and Y and Y and Z, but not between X and Z. But conditional independence is a symmetric relationship, so how could we **orient** those edges, give them direction? Well, to rehearse a point from the last section, there are only four possible directed graphs corresponding to that undirected graph:

- $X \rightarrow Y \rightarrow Z$ (a chain);
- $X \leftarrow Y \leftarrow Z$ (the other chain);
- $X \leftarrow Y \rightarrow Z$ (a fork on Y);
- $X \rightarrow Y \leftarrow Z$ (a collision at Y)

With the fork or either chain, we have $X \perp\!\!\!\perp Z|Y$. On the other hand, with the collider we have $X \not\perp\!\!\!\perp Z|Y$. Thus $X \not\perp\!\!\!\perp Z|Y$ if and only if there is a collision at Y. By testing for *this* conditional dependence, we can either definitely orient the edges, or rule out an orientation. If $X - Y - Z$ is just a subgraph of a larger graph, we can still identify it as a collider if $X \not\perp\!\!\!\perp Z|\{Y, S\}$ for *all* collections of nodes S (not including X and Z themselves, of course).

With more nodes and edges, we can **induce** more orientations of edges by consistency with orientations we get by identifying colliders. For example, suppose we know that X, Y, Z is either a chain or a fork on Y. If we learn that $X \rightarrow Y$, then the triple *cannot* be a fork, and must be the chain $X \rightarrow Y \rightarrow Z$. So orienting the $X - Y$ edge induces an orientation of the $Y - Z$ edge. We can also sometimes orient edges through background knowledge; for instance we might know that Y comes later in time than X, so if there is an edge between them it *cannot* run

[6] But see Janzing (2007); Hoyer *et al.* (2009) for some ideas on how you could do it if you're willing to make some extra assumptions. The basic idea of these papers is that the distribution of effects given causes should be simpler, in some sense, than the distribution of causes given effects.

[7] Possibly empty: conditioning on the empty set of variables is the same as not conditioning at all.

[8] "No causation without association", as it were.

from Y to X.[9] We can eliminate other edges based on similar sorts of background knowledge: males tend to be heavier than females, but changing weight does not change sex, so there can't be an edge (or even a directed path!) from weight to sex, though there could be one the other way around.

To sum up, we can rule out an edge between X and Y whenever we can make them independent by conditioning on other variables; and when we have an $X - Y - Z$ pattern, we can identify colliders by testing whether X and Z are dependent given Y. Having oriented the arrows going into colliders, we induce more orientations of other edges.

Putting these three things — edge elimination by testing, collider finding, and inducing orientations — gives the most basic causal discovery procedure, the SGS (Spirtes-Glymour-Scheines) algorithm (Spirtes *et al.*, 2001, §5.4.1, p. 82). This assumes:

1. The data-generating distribution has the causal Markov property on a graph G.
2. The data-generating distribution is faithful to G.
3. Every member of the population has the same distribution.
4. All relevant variables are in G.
5. There is only *one* graph G to which the distribution is faithful.

Abstractly, the algorithm works as follows:

- Start with a complete undirected graph on all p variables, with edges between all nodes.
- For each pair of variables X and Y, and each set of other variables S, see if $X \perp\!\!\!\perp Y|S$; if so, remove the edge between X and Y.
- Find colliders by checking for conditional dependence; orient the edges of colliders.
- Try to orient undirected edges by consistency with already-oriented edges; do this recursively until no more edges can be oriented.

Pseudo-code is in §22.7.

Call the result of the SGS algorithm $\widehat{G}$. If all of the assumptions above hold, and the algorithm is correct in its guesses about when variables are conditionally independent, then $\widehat{G} = G$. In practice, of course, conditional independence guesses are really statistical tests based on finite data, so we should write the output as

[9] Some have argued, or at least entertained the idea, that the logic here is backwards: rather than order in time constraining causal relations, causal order *defines* time order. (Versions of this idea are discussed by, inter alia, Russell (1927); Wiener (1961); Reichenbach (1956); Pearl (2009b); Janzing (2007) makes a related suggestion). Arguably then using order in time to orient edges in a causal graph begs the question, or commits the fallacy of *petitio principii*. But of course every syllogism does, so this isn't a distinctively *statistical* issue. (Take the classic: "All men are mortal; Socrates is a man; therefore Socrates is mortal." How can we know that *all* men are mortal until we know about the mortality of this particular man, Socrates? Isn't this just like asserting that tomatoes and peppers must be poisonous, because they belong to the nightshade family of plants, all of which are poisonous?) While these philosophical issues are genuinely fascinating, this footnote has gone on long enough, and it is time to return to the main text.

$\widehat{G}_n$, to indicate that it is based on only n samples. If the conditional independence test is consistent, then

$$\lim_{n\to\infty} \Pr\left(\widehat{G}_n \neq G\right) = 0 \tag{22.7}$$

In other words, the SGS algorithm converges in probability on the correct causal structure; it is consistent for all graphs G. Of course, at finite n, the probability of error — of having the wrong structure — is (generally!) not zero, but this just means that, like any statistical procedure, we cannot be absolutely certain that it's not making a mistake.

One consequence of the independence tests making errors on finite data can be that we fail to orient some edges — perhaps we missed some colliders. These unoriented edges in $\widehat{G}_n$ can be thought of as something like a confidence region — they have *some* orientation, but multiple orientations are all compatible with the data.[10] As more and more edges get oriented, the confidence region shrinks.

If the fifth assumption above fails to hold, then there are multiple graphs G to which the distribution is faithful. This is just a more complicated version of the difficulty of distinguishing between the graphs $X \to Y$ and $X \leftarrow Y$. All the graphs in the equivalence class may have some arrows in common; in that case the SGS algorithm will identify those arrows. If some edges differ in orientation across the equivalence class, SGS will not orient them, even in the limit. In terms of the previous paragraph, the confidence region never shrinks to a single point, just because the data doesn't provide the information needed to do this. The graph is only partially identified.

If there *are* unmeasured relevant variables, we can get not just un-oriented edges, but actually arrows pointing in both directions. This is an excellent sign that some basic assumption is being violated.

22.4.1 The PC Algorithm

The SGS algorithm is statistically consistent, but very computationally inefficient; the number of tests it does grows exponentially in the number of variables p. This is the worst-case complexity for *any* consistent causal-discovery procedure, but this algorithm just proceeds immediately to the worst case, not taking advantage of any possible short-cuts.

Since it's enough to find *one* S making X and Y independent to remove their edge, one obvious short-cut is to do the tests in some order, and skip unnecessary tests. On the principle of doing the easy work first, the revised edge-removal step would look something like this:

- For each X and Y, see if $X \perp\!\!\!\perp Y$; if so, remove their edge.
- For each X and Y which are still connected, and each third variable Z connected to X or Y, see if $X \perp\!\!\!\perp Y|Z$; if so, remove the edge between X and Y.

[10] I say "multiple orientations" rather than "all orientations", because picking a direction for one edge might induce an orientation for others.

- For each X and Y which are still connected, and each third and fourth variables Z_1 and Z_2 both connected to X or both connected to Y, see if $X \perp\!\!\!\perp Y | Z_1, Z_2$; if so, remove the edge between X and Y.
- ...
- For each X and Y which are still connected at the kth stage, see if there are k variables $Z_1, Z_2, \ldots Z_k$ all connected to X or all connected to Y where $X \perp\!\!\!\perp Y | \{Z_1, \ldots Z_k\}$; if so, remove, the edge between X and Y.
- ...
- Stop when $k = p - 2$.

If all the tests are done correctly, this will give the same result as the SGS procedure (Exercise 22.4). And if some of the tests give erroneous results, conditioning on a small number of variables will tend to be more reliable than conditioning on more (why?).

We can be even more efficient, however. If $X \perp\!\!\!\perp Y | S$ for any S at all, then $X \perp\!\!\!\perp Y | S'$, where all the variables in S' are adjacent to X or Y (or both) (Exercise 22.3). To see the sense of this, suppose that there is a single long directed path running from X to Y. If we condition on any of the variables along the chain, we make X and Y independent, but we could always move the point where we block the chain to be either right next to X or right next to Y. So when we are trying to remove edges and make X and Y independent, we only need to condition on variables which are still connected to X and Y, not ones in totally different parts of the graph.

This then gives us the PC[11] algorithm (Spirtes *et al.* 2001, §5.4.2, pp. 84–88; see also §22.7). It works exactly like the SGS algorithm, except for the edge-removal step, where it tries to condition on as few variables as possible (as above), and only conditions on adjacent variables. The PC algorithm has the same assumptions as the SGS algorithm, and the same consistency properties, but generally runs much faster, and does many fewer statistical tests. It should be the default algorithm for attempting causal discovery.

22.4.2 Causal Discovery with Hidden Variables

Suppose that the set of variables we measure is *not* causally sufficient. Could we at least discover this? Could we possibly get hold of *some* of the causal relationships? Algorithms which can do this exist (e.g., the CI and FCI algorithms of Spirtes *et al.* (2001, ch. 6)), but they require considerably more graph-fu. (The RFCI algorithm (Colombo *et al.*, 2012) is a modern, fast successor to FCI.) The results of these algorithms can succeed in removing *some* edges between observable variables, and definitely orienting some of the remaining edges. If there are actually no latent common causes, they end up acting like the SGS or PC algorithms.

[11] Peter-Clark

Partial identification of effects

When all relevant variables are observed, all effects are identified within one graph; partial identification happens because multiple graphs are equivalent. When some variables are not observed, we may have to use the identification strategies to get at the same effect. In fact, the same effect may be identified in one graph and not identified in another, equivalent graph. This is, again, unfortunate, but when it happens it needs to be admitted.

22.4.3 On Conditional Independence Tests

The abstract algorithms for causal discovery assume the existence of consistent tests for conditional independence. The implementations known to me mostly assume either that variables are discrete (so that one can basically use the χ^2 test), or that they are continuous, Gaussian, and linearly related (so that one can test for vanishing partial correlations), though the `pcalg` package does allow users to provide their own conditional independence tests as arguments. It bears emphasizing that these restrictions are *not* essential. As soon as you have a consistent independence test, you are, in principle, in business. In particular, consistent *non-parametric* tests of conditional independence would work perfectly well. An interesting example of this is the paper by Chu and Glymour (2008), on finding causal models for the time series, assuming additive but non-linear models.

22.5 Software and Examples

The PC and FCI algorithms are implemented in the stand-alone Java program `Tetrad` (`http://www.phil.cmu.edu/projects/tetrad/`). They are also implemented in the `pcalg` package on CRAN (Kalisch *et al.*, 2010, 2012). This package also includes functions for calculating the effects of interventions from fitted graphs, assuming linear models. The documentation for the package is somewhat confusing; rather see Kalisch *et al.* (2012) for a tutorial introduction.

[[TODO: Cleanup output from the package]]

It's worth going through how `pcalg` works[12]. The code is designed to take advantage of the modularity and abstraction of the PC algorithm itself; it separates actually finding the graph completely from performing the conditional independence test, which is rather a function the user supplies. (Some common ones are built in.) For reasons of computational efficiency, in turn, the conditional independence tests are set up so that the user can just supply a set of sufficient statistics, rather than the raw data.

Let's walk through an example[13], using the `mathmarks` data set. This contains

[12] A word about installing the package: you'll need the package `Rgraphviz` for drawing graphs, which is hosted not on CRAN (like `pcalg`) but on BioConductor. Try installing it, and its dependencies, before installing `pcalg`. See `http://www.bioconductor.org/packages/release/bioc/html/Rgraphviz.html` for help on installing `Rgraphviz`.

[13] After Spirtes *et al.* (2001, §6.12, pp. 152–154).

grades ("marks") from 88 university students in five mathematical subjects, algebra, analysis, mechanics, statistics and vectors. All five variables are positively correlated with each other.

```
library(pcalg)
library(SMPracticals)
data(mathmarks)
suffStat <- list(C=cor(mathmarks),n=nrow(mathmarks))
pc.fit <- pc(suffStat, indepTest=gaussCItest, p=ncol(mathmarks),alpha=0.005)
```

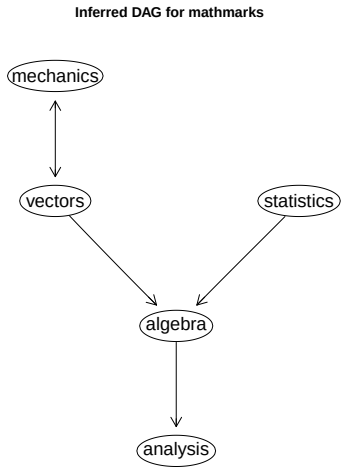
This uses a Gaussian (-and-linear) test for conditional independence, `gaussCItest`, which is built into the `pcalg` package. Basically, it hopes to test whether $X \perp\!\!\!\perp Y|Z$ by testing whether the partial correlation of X and Y given Z is close to zero. These partial correlations can all be calculated from the correlation matrix, so the line before creates the sufficient statistics needed by `gaussCItest` — the matrix of correlations and the number of data points. We also have to tell `pc` how many variables there are, and what significance level to use in the test (here, 0.5%).

Before going on, I encourage you to run `pc` as above, but with `verbose=TRUE`, and to study the output.

Figure 22.3 shows the resulting DAG. If we take it seriously, it says that grades in analysis are driven by grades in algebra, while algebra in turn is driven by statistics and vectors. While one could make up stories for why this would be so (perhaps something about the curriculum?), it seems safer to regard this as a warning against *blindly* trusting any algorithm —- a key assumption of the PC algorithm, after all, is that there are no unmeasured but causally-relevant variables, and it is easy to believe these are violated. For instance, while *knowledge of* different mathematical fields may be causally linked (it would indeed be hard to learn much mechanics without knowing about vectors), test scores are only imperfect measurements of knowledge.

The size of the test may seem low, but remember we are doing a lot of tests:

```
summary(pc.fit)
## Object of class 'pcAlgo', from Call:
## pc(suffStat = suffStat, indepTest = gaussCItest, alpha = 0.005,
##     p = ncol(mathmarks))
##
## Nmb. edgetests during skeleton estimation:
## ===========================================
## Max. order of algorithm:  3
## Number of edgetests from m = 0 up to m = 3 :  20 38 10 0
##
## Graphical properties of skeleton:
## ================================
## Max. number of neighbours:  2 at node(s) 2
## Avg. number of neighbours:  1
##
## Adjacency Matrix G:
##   1 2 3 4 5
## 1 . 1 . . .
## 2 1 . 1 . .
## 3 . . . 1 .
```

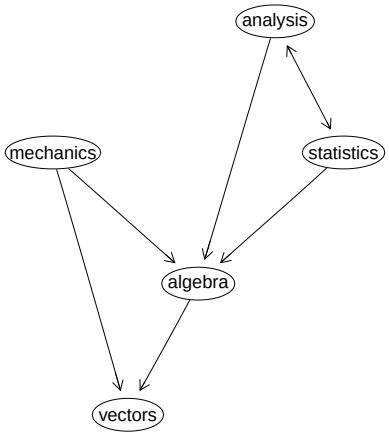



```
library(Rgraphviz)
plot(pc.fit,labels=colnames(mathmarks),main="Inferred DAG for mathmarks")
```

Figure 22.3 DAG inferred by the PC algorithm from the `mathmarks` data. Two-headed arrows, like undirected edges, indicate that the algorithm was unable to orient the edge. (It is obscure why `pcalg` sometimes gives an edge it cannot orient no heads and sometimes two.)

```
## 4 . . . . .
## 5 . . 1 . .
```

This tells us that it considered going up to conditioning on three variables (the maximum possible, since there are only five variables), that it did twenty tests of unconditional independence, 31 tests where it conditioned on one variable, four tests where it conditioned on two, and none where it conditioned on three. This 55 tests in all, so a simple Bonferroni correction suggests the over-all size is $55 \times 0.005 = 0.275$. This is probably pessimistic (the Bonferroni correction typically is). Setting $\alpha = 0.05$ gives a somewhat different graph (Figure 22.4).




```
plot(pc(suffStat, indepTest=gaussCItest, p=ncol(mathmarks),alpha=0.05),
  labels=colnames(mathmarks),main="")
```

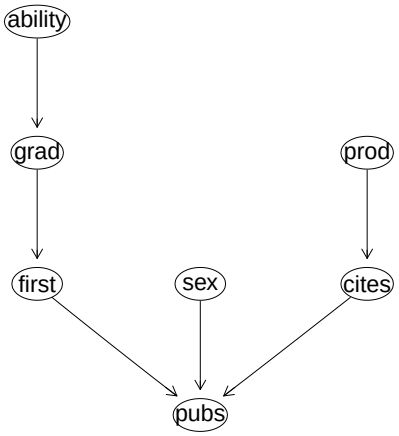
Figure 22.4 Inferred DAG when the size of the test is 0.05.

For a second example[14], let's use some data on academic productivity among psychologists. The two variables of ultimate interest were the publication (`pubs`) and citation (`cites`) rates, with possible measured causes including `ability` (basically, standardized test scores), graduate program quality `grad` (basically, the program's national rank), the quality of the psychologist's first job, `first`, a measure of productivity `prod`, and sex. There were 162 subjects, and while the actual data isn't reported, the correlation matrix is.

```
psychs
##         ability grad prod first   sex cites pubs
## ability    1.00 0.62 0.25  0.16 -0.10  0.29 0.18
## grad       0.62 1.00 0.09  0.28  0.00  0.25 0.15
## prod       0.25 0.09 1.00  0.07  0.03  0.34 0.19
## first      0.16 0.28 0.07  1.00  0.10  0.37 0.41
## sex       -0.10 0.00 0.03  0.10  1.00  0.13 0.43
## cites      0.29 0.25 0.34  0.37  0.13  1.00 0.55
## pubs       0.18 0.15 0.19  0.41  0.43  0.55 1.00
```

The model found by `pcalg` is fairly reasonable-looking (Figure 22.5). Of course, the linear-and-Gaussian assumption has no particular support here, and there is at least one variable for which it must be wrong (which?), but unfortunately with just the correlation matrix we cannot go further.

[14] Following Spirtes *et al.* (2001, §5.8.1, pp. 98–102).




```
plot(pc(list(C=psychs,n=162),indepTest=gaussCItest,p=7,alpha=0.01),
  labels=colnames(psychs),main="")
```

Figure 22.5 Causes of academic success among psychologists. The arrow from citations *to* publications is a bit odd, but not impossible — people who get cited more might get more opportunities to do research and so to publish.

22.6 Limitations on Consistency of Causal Discovery

There are some important limitations to causal discovery algorithms (Spirtes *et al.*, 2001, §12.4). They are *universally* consistent: for all causal graphs G,[15]

$$\lim_{n\to\infty} \Pr\left(\widehat{G}_n \neq G\right) = 0 \tag{22.8}$$

The probability of getting the graph wrong can be made arbitrarily small by using enough data. However, this says nothing about *how much* data we need to achieve a given level of confidence, i.e., the *rate* of convergence. *Uniform* consistency would mean that we could put a bound on the probability of error as a function of n which did not depend on the true graph G. Robins *et al.* (2003) proved that *no* uniformly-consistent causal discovery algorithm can exist. The issue, basically, is that the Adversary could make the convergence in Eq. 22.8 arbitrarily slow by selecting a distribution which, while faithful to G, came *very close* to being unfaithful, making some of the dependencies implied by the graph arbitrarily small. For any given dependence strength, there's some amount of data which will let us recognize it with high confidence, but the Adversary can make the required data size as large as he likes by weakening the dependence, without ever setting it to zero[16].

The upshot is that so *uniform*, *universal* consistency is out of the question; we can be *universally* consistent, but without a uniform rate of convergence; or we can converge *uniformly*, but only on some less-than-universal class of distributions. These might be ones where all the dependencies which do exist are not too weak (and so not too hard to learn reliably from data), or the number of true edges is not too large (so that if we haven't seen edges yet they probably don't exist; Janzing and Herrmann, 2003; Kalisch and Bühlmann, 2007).

It's worth emphasizing that the Robins *et al.* (2003) no-uniform-consistency result applies to *any* method of discovering causal structure from data. Invoking human judgment, Bayesian prior distributions over possible causal structures, etc., etc., won't get you out of it.

22.7 Pseudo-code for the SGS Algorithm[17]

When you see a loop, assume that it gets entered at least once. "Replace" in the sub-functions always refers to the input graph.

```
SGS = function(set of variables V) {
    Ĝ = colliders(prune( complete undirected graph on V))
    until (Ĝ == G′) {
        Ĝ = G′
```

[15] If the true distribution is faithful to multiple graphs, then we should read G as their equivalence class, which has some undirected edges.

[16] See §18.4 for a more quantitative statement of how the required sample size relates to non-parametric measures of the strength of dependence.

[17] This section may be omitted on first (and maybe even second) reading.

```
            G' = orient(Ĝ)
    }
    return(Ĝ)
}

prune = function(G) {
    for each A, B ∈ V {
        for each S ⊆ V \ {A, B} {
            if A ⫫ B|S { G = G \ (A − B) }
        }
    }
    return(G)
}

collliders = function(G) {
    for each (A − B) ∈ G {
        for each (B − C) ∈ G {
            if (A − C) ∉ G {
                collision = TRUE
                for each S ⊂ B ∩ V \ {A, C} {
                    if A ⫫ C|S { collision = FALSE }
                }
                if (collision) { replace (A − B) with (A → B), (B − C) with (B ← C) }
            }
        }
    }
    return(G)
}

orient = function(G) {
    if ((A → B) ∈ G & (B − C) ∈ G & (A − C) ∉ G) { replace (B − C) with (B → C) }
    if ((directed path from A to B)∈ G & (A − B) ∈ G) { replace (A − B) with (A → B) }
    return(G)
}
```

22.8 Further Reading

The best single reference on causal discovery algorithms remains Spirtes *et al.* (2001). A lot of work has been done in recent years by the group centered around ETH-Zürich, beginning with Kalisch and Bühlmann (2007), connecting this to modern statistical concerns about sparse effects and high-dimensional modeling.

As already mentioned, the best reference on partial identification is Manski (2007). Partial identification of causal effects due to multiple equivalent DAGs is considered in Maathuis *et al.* (2009), along with efficient algorithms for linear

systems, which are applied in Maathuis *et al.* (2010), and implemented in the `pcalg` package as `ida`.

Discovery is possible for directed cyclic graphs, though since it's harder to understand what such models mean, it is less well-developed. Important papers on this topic include Richardson (1996) and Lacerda *et al.* (2008).

Exercises

22.1 Prove that, assuming faithfulness, a three-variable chain and a three-variable fork imply exactly the same set of dependence and independence relations, but that these are different from those implied by a three-variable collider. Are any implications common to chains, forks, and colliders? Could colliders be distinguished from chains and forks without assuming faithfulness?

22.2 Prove that if X and Y are not parent and child, then either $X \perp\!\!\!\perp Y$, or there exists a set of variables S such that $X \perp\!\!\!\perp Y|S$. *Hint:* start with the Markov property, that any X is independent of all its non-descendants given its parents, and consider separately the cases where Y a descendant of X and those where it is not.

22.3 Prove that if $X \perp\!\!\!\perp Y|S$ for some set of variables S, then $X \perp\!\!\!\perp Y|S'$, where every variable in S' is a neighbor of X or Y.

22.4 Prove that the graph produced by the edge-removal step of the PC algorithm is exactly the same as the graph produced by the edge-removal step of the SGS algorithm. *Hint:* SGS removes the edge between X and Y when $X \perp\!\!\!\perp Y|S$ for *even one* set S.

22.5 When, exactly, does $\mathbb{E}[Y \mid X, Z] = \mathbb{E}[Y \mid Z]$ imply $Y \perp\!\!\!\perp X|Z$?

22.6 Would the SGS algorithm work on a non-causal, merely-probabilistic DAG? If so, in what sense is it a *causal* discovery algorithm? If not, why not?

22.7 Describe how to use bandwidth selection as a conditional independence test.

22.8 Read Kalisch *et al.* (2012) and write a conditional independence test function based on bandwidth selection (§14.5). Check that your test gives the right size when run on simulated cases where you know the variables are conditionally independent. Check that your test function works with `pcalg::pc`.

Part IV

Dependent Data

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

23

Time Series

So far, we have assumed that all data points are pretty much independent of each other. In the chapters on regression, we assumed that each Y_i was independent of every other, given its X_i, and we often assumed that the X_i were themselves independent. In Parts II and III, we allowed for arbitrarily complicated dependence between the *variables*, but each multivariate *data-point* was assumed to be generated independently. We will now relax this assumption, and see what sense we can make of dependent data.

23.1 What Time Series Are

The simplest form of dependent data are time series, which are just what they sound like: a series of values recorded over time. The most common version of this, in statistical applications, is to have measurements of a variable or variables X at equally-spaced time-points starting from t, written say $X_t, X_{t+h}, X_{t+2h}, \ldots$, or $X(t), X(t+h), X(t+2h), \ldots$. Here h, the amount of time between observations, is called the "sampling interval", and $1/h$ is the "sampling frequency" or "sampling rate".

Figure 23.1 shows two fairly typical time series. One of them is actual data (the number of lynxes trapped each year in a particular region of Canada); the other is the output of a purely artificial model. (Without the labels, it might not be obvious which one was which.) The core idea of time series analysis is one which we're already familiar with from the rest of statistics: we regard the actual time series we see as one realization of some underlying, partially-random ("stochastic") process, which generated the data. We use the data to make guesses ("inferences") about the process, and want to make *reliable* guesses while being clear about the uncertainty involved. The complication is that each observation is dependent on all the other observations; in fact it's usually this dependence that we want to draw inferences about.

Other kinds of time series

One sometimes encounters irregularly-sampled time series, $X(t_1), X(t_2), \ldots$, where $t_i - t_{i-1} \neq t_{i+1} - t_i$. This is mostly an annoyance, unless the observation times are somehow dependent on the values. Continuously-observed processes are rarer — especially now that digital sampling has replaced analog measurement in so many applications. (It is more common to model the process as evolving continuously

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

```
logistic.map <- function(x, r = 4) {
    r * x * (1 - x)
}
logistic.iteration <- function(n, x.init, r = 4) {
    x <- vector(length = n)
    x[1] <- x.init
    for (i in 1:(n - 1)) {
        x[i + 1] <- logistic.map(x[i], r = r)
    }
    return(x)
}
x <- logistic.iteration(1000, x.init = runif(1))
y <- x + rnorm(1000, mean = 0, sd = 0.05)
```

CODE EXAMPLE 31: *Code defining our synthetic data set. Why is this "logistic"?*

in time, but observe it at discrete times.) We skip both of these in the interest of space.

Regular, irregular or continuous time series all record the same variable at every moment of time. An alternative is to just record the sequence of times at which some event happened; this is called a "point process". More refined data might record the time of each event and its type — a "marked point process". Point processes include very important kinds of data (e.g., earthquakes), but they need special techniques, and we'll skip them (though see §23.12).

Notation

For a regularly-sampled time series, it's convenient not to have to keep writing the actual time, but just the position in the series, as $X_1, X_2, \ldots$, or $X(1), X(2), \ldots$. This leads to a useful short-hand, that $X_{i:j} = (X_i, X_{i+1}, \ldots X_{j-1}, X_j)$, a whole block of time; some people write X_i^j with the same meaning.

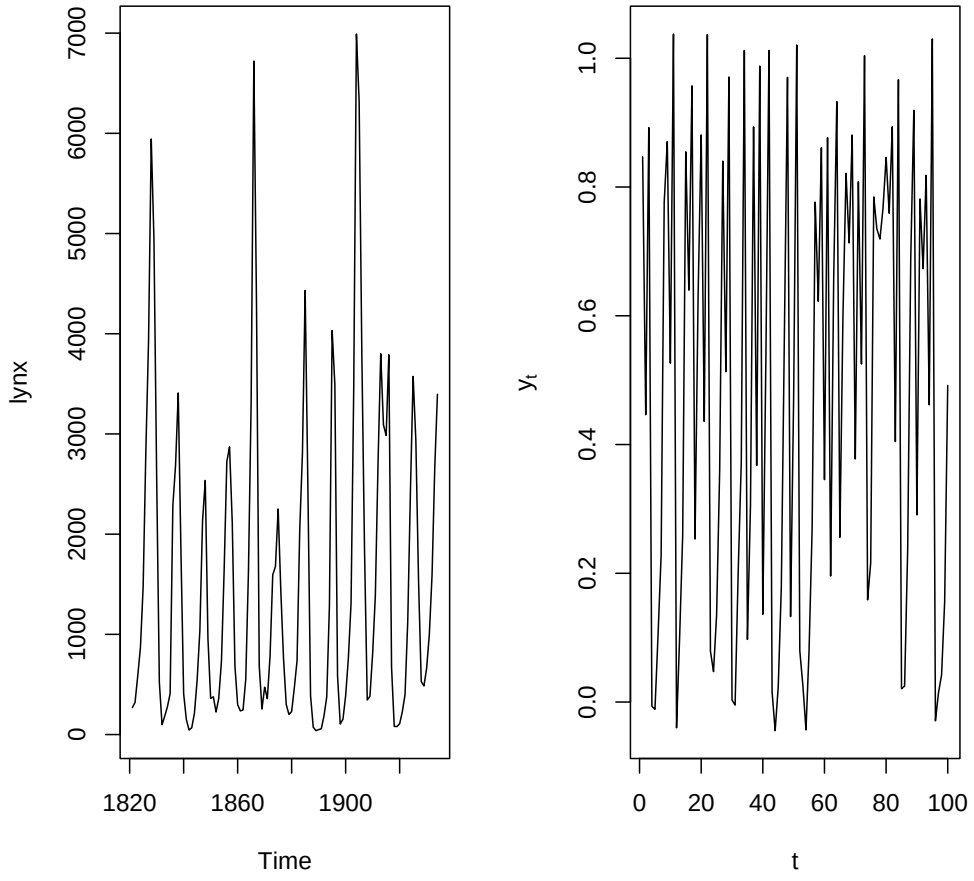
23.2 Stationarity

In our old IID world, the distribution of each observation is the same as the distribution of every other data point. It would be nice to have something like this for time series. The property is called **stationarity**, which doesn't mean that the *value* of the time series never changes, but that its *distribution* doesn't.

More precisely, a time series is **strictly stationary** or **strongly stationary** when $X_{1:k}$ and $X_{t:t+k-1}$ have the same distribution, for all k and t — the distribution of blocks of length k is **time-invariant**. Again, this doesn't mean that every block of length k has the same value, just that it has the same distribution of values.

If there is strong or strict stationarity, there should be **weak** or **loose** (or **wide-sense**) stationarity, and there is. All it requires is that $\mathbb{E}[X_1] = \mathbb{E}[X_t]$, and that $\mathrm{Cov}[X_1, X_k] = \mathrm{Cov}[X_t, X_{t+k-1}]$. (Notice that it's not dealing with whole blocks of time any more, just one or two time-points.)

Strong stationarity implies weak stationarity, but not, in general, the other




```
par(mfrow = c(1, 2))
plot(lynx)
plot(y[1:100], xlab = "t", ylab = expression(y[t]), type = "l")
par(mfrow = c(1, 1))
```

Figure 23.1 Left: annual number of trapped lynxes in the Mackenzie River region of Canada. Right: a toy dynamical model, simulated from Code Example 31.

way around, hence the names. It may not surprise you to learn that strong and weak stationarity coincide when X_t is a Gaussian process, but not, in general, otherwise. You can prove all the claims in this paragraph in Exercise 23.1.

You should convince yourself that an IID sequence is strongly stationary.

23.2.1 Autocorrelation

Time series are **serially dependent**: X_t is in general dependent on all earlier values in time, and on all later ones. Typically, however, there is **decay of dependence** (sometimes called **decay of correlations**): X_t and X_{t+h} become more and more nearly independent as $h \to \infty$. The oldest way of measuring this is the **autocovariance**,

$$\gamma(h) = \mathrm{Cov}\left[X_t, X_{t+h}\right] \tag{23.1}$$

which is well-defined just when the process is weakly stationary. We could equally well use the **autocorrelation**,

$$\rho(h) = \frac{\mathrm{Cov}\left[X_t, X_{t+h}\right]}{\mathbb{V}\left[X_t\right]} = \frac{\gamma(h)}{\gamma(0)} \tag{23.2}$$

again using stationarity to simplify the denominator.

As I said, for most time series $\gamma(h) \to 0$ as h grows. Of course, $\gamma(h)$ could be exactly zero while X_t and X_{t+h} are strongly dependent. Figure 23.2 shows the autocorrelation functions (ACFs) of the lynx data and the simulation model; the correlation for the latter is basically never distinguishable from zero, which doesn't accord at all with the visual impression of the series. Indeed, we can confirm that *something* is going on the series by the simple device of plotting X_{t+1} against X_t (Figure 23.3). More general measures of dependence would include looking at the Spearman rank-correlation of X_t and X_{t+h}, or quantities like mutual information.

Autocorrelation is important for four reasons, however. First, because it is the oldest measure of serial dependence, it has a "large installed base": everybody knows about it, they use it to communicate, and they'll ask you about it. Second, in the rather special case of Gaussian processes, it really does tell us everything we need to know. Third, in the somewhat less special case of linear prediction, it tells us everything we need to know. Fourth and finally, it plays an important role in a crucial theoretical result, which we'll go over next.

23.2.2 The Ergodic Theorem

With IID data, the ultimate basis of all our statistical inference is the law of large numbers, which told us that

$$\frac{1}{n}\sum_{i=1}^{n} X_i \to \mathbb{E}\left[X_1\right] \tag{23.3}$$

For complicated historical reasons, the corresponding result for time series is called the **ergodic theorem**[1]. The most general and powerful versions of it are

[1] In the late 1800s, the physicist Ludwig Boltzmann needed a word to express the idea that if you took an isolated system at constant energy and let it run, any one trajectory, continued long enough, would be representative of the system as a whole. Being a highly-educated nineteenth century German-speaker, Boltzmann knew far too much ancient Greek, so he called this the "ergodic property", from *ergon* "energy, work" and *hodos* "way, path". The name stuck.

quite formidable, and have very subtle proofs, but there is a simple version which gives the flavor of them all, and is often useful enough.

23.2.2.1 The World's Simplest Ergodic Theorem

Suppose X_t is weakly stationary, and that

$$\sum_{h=0}^{\infty} |\gamma(h)| = \gamma(0)\tau < \infty \tag{23.4}$$

(Remember that $\gamma(0) = \mathbb{V}[X_t]$.) The quantity τ is called the **correlation time**, or **integrated autocorrelation time**.

Now consider the average of the first n observations,

$$\overline{X}_n = \frac{1}{n}\sum_{t=1}^{n} X_t \tag{23.5}$$

This **time average** is a random variable. Its expectation value is

$$\mathbb{E}\left[\overline{X}_n\right] = \frac{1}{n}\sum_{t=1}^{n}\mathbb{E}[X_t] = \mathbb{E}[X_1] \tag{23.6}$$

because the mean is stationary. What about its variance?

$$\begin{aligned}
\mathbb{V}\left[\overline{X}_n\right] &= \mathbb{V}\left[\frac{1}{n}\sum_{t=1}^{n} X_t\right] && (23.7)\\
&= \frac{1}{n^2}\left[\sum_{t=1}^{n}\mathbb{V}[X_t] + 2\sum_{t=1}^{n}\sum_{s=t+1}^{n}\mathrm{Cov}[X_t, X_s]\right] && (23.8)\\
&= \frac{1}{n^2}\left[n\mathbb{V}[X_1] + 2\sum_{t=1}^{n}\sum_{s=t+1}^{n}\gamma(s-t)\right] && (23.9)\\
&\leq \frac{1}{n^2}\left[n\gamma(0) + 2\sum_{t=1}^{n}\sum_{s=t+1}^{n}|\gamma(s-t)|\right] && (23.10)\\
&\leq \frac{1}{n^2}\left[n\gamma(0) + 2\sum_{t=1}^{n}\sum_{h=1}^{n}|\gamma(h)|\right] && (23.11)\\
&\leq \frac{1}{n^2}\left[n\gamma(0) + 2\sum_{t=1}^{n}\sum_{h=1}^{\infty}|\gamma(h)|\right] && (23.12)\\
&= \frac{n\gamma(0)(1+2\tau)}{n^2} && (23.13)\\
&= \frac{\gamma(0)(1+2\tau)}{n} && (23.14)
\end{aligned}$$

Eq. 23.9 uses stationarity again, and then Eq. 23.13 uses the assumption that the correlation time τ is finite.

Since $\mathbb{E}\left[\overline{X_n}\right] = \mathbb{E}\left[X_1\right]$, and $\mathbb{V}\left[\overline{X}_n\right] \to 0$, we have that

$$overline X_n \to \mathbb{E}\left[X_1\right] \tag{23.15}$$

exactly as in the IID case. ("Time averages converge on expected values.") In fact, we can say a bit more. Remember Chebyshev's inequality: for any random variable Z,

$$\Pr\left(|Z - \mathbb{E}\left[Z\right]| > \epsilon\right) \leq \frac{\mathbb{V}\left[Z\right]}{\epsilon^2} \tag{23.16}$$

so

$$\Pr\left(|\overline{X}_n - \mathbb{E}\left[X_1\right]| > \epsilon\right) \leq \frac{\gamma(0)(1+2\tau)}{n\epsilon^2} \tag{23.17}$$

which goes to zero as n grows for any given ϵ.

You may wonder whether the condition that $\sum_{h=0}^{\infty} |\gamma(h)| < \infty$ is as weak as possible. It turns out that it can in fact be weakened to just $\lim_{n\to\infty} \frac{1}{n}\sum_{h=0}^{n} \gamma(h) = 0$, as indeed the proof above might suggest.

The argument above can actually be extended to some non-stationary processes; see Exercise 23.8.

23.2.2.2 Rate of Convergence

If the X_t were all IID, or even just uncorrelated, we would have $\mathbb{V}\left[\overline{X}_n\right] = \gamma(0)/n$ exactly. Our bound on the variance is larger by a factor of $(1+2\tau)$, which reflects the influence of the correlations. Said another way, we can more or less pretend that instead of having n correlated data points, we have $n/(1+2\tau)$ independent data points, that $n/(1+2\tau)$ is our **effective sample size**[2]

Generally speaking, dependence between observations reduces the effective sample size, and the stronger the dependence, the greater the reduction. (For an extreme example, consider the situation where X_1 is randomly drawn, but thereafter $X_{t+1} = X_t$.) In more complicated situations, finding the effective sample size is itself a tricky undertaking, but it's often got this general flavor.

23.2.2.3 Why Ergodicity Matters

The ergodic theorem is important, because it tells us that a single long time series becomes representative of the whole data-generating process, just the same way that a large IID sample becomes representative of the whole population or distribution. We can therefore actually learn about the process from empirical data.

Strictly speaking, we have established that time-averages converge on expectations only for X_t itself. This doesn't directly address what happens for a transformation $f(X_t)$ where the function f is non-linear. It might be that $f(X_t)$ doesn't have a finite correlation time even though X_t does, or indeed vice versa. This is annoying; we don't want to have to go through the analysis of the last section for every different function we might want to calculate.

[2] Some people like to define the correlation time as, in this notation, $1 + 2\tau$ for just this reason.

When people say that the whole process is **ergodic**, they roughly speaking mean that

$$\frac{1}{n}\sum_{t=1}^{n} f(X_{t:t+k-1}) \rightarrow \mathbb{E}\left[f(X_{1:k})\right] \tag{23.18}$$

for any reasonable function f. This is (again very roughly) equivalent to

$$\frac{1}{n}\sum_{t=1}^{n} \Pr\left(X_{1:k} \in A, X_{t:t+l-1} \in B\right) \rightarrow \Pr\left(X_{1:k} \in A\right)\Pr\left(X_{1:l} \in B\right) \tag{23.19}$$

which is a kind of asymptotic independence-on-average[3]

If our data source is ergodic, then what Eq. 23.18 tells us is that sample averages of any reasonable function are representative of expectation values, which is what we need to be in business statistically. This in turn is basically implied by stationarity.[4] What does this let us do?

23.3 Markov Models

For this section, we'll assume that X_t comes from a stationary, ergodic time series. So for any reasonable function f, the time-average of $f(X_t)$ converges on $\mathbb{E}\left[f(X_1)\right]$. Among the "reasonable" functions are the indicators, so

$$\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}_A(X_t) \rightarrow \Pr\left(X_1 \in A\right) \tag{23.20}$$

[3] Here's a sketch of a less rough statement. Instead of working with X_t, work with the whole future trajectory $Y_t = (X_t, X_{t+1}, X_{t+2}, \ldots)$. Now the dynamics, the rule which moves us into the future, can be summed up in a very simple, and deterministic, operation T: $Y_{t+1} = TY_t = (X_{t+1}, X_{t+2}, X_{t+3}, \ldots)$. A set of trajectories is **invariant** if it is left unchanged by T: for every $y \in A$, there is another y' in A where $Ty' = y$. A process is **ergodic** if every invariant set either has probability 0 or probability 1. What this means is that (almost) all trajectories generated by an ergodic process belong to a single invariant set, and they all wander from every part of that set to every other part — they are **metrically transitive**. (Because: no smaller set with any probability is invariant.) Metric transitivity, in turn, is equivalent, assuming stationarity, to $n^{-1}\sum_{t=0}^{n-1} \Pr\left(Y \in A, T^t Y \in B\right) \rightarrow \Pr\left(Y \in A\right)\Pr\left(Y \in B\right)$. From metric transitivity follows Birkhoff's "individual" ergodic theorem, that $n^{-1}\sum_{t=0}^{n-1} f(T^t Y) \rightarrow \mathbb{E}\left[f(Y)\right]$, with probability 1. Since a function of the trajectory can be a function of a block of length k, we get Eq. 23.18. — These definitions, and the proofs of the associated claims, are all pretty standard in ergodic theory; Gray (2009) is a good source.

[4] Another sketch of a less rough statement: Use Y again for whole trajectories. Every stationary distribution for Y can be written as a mixture of stationary and ergodic distributions, rather as we wrote complicated distributions as mixtures of simple Gaussians in Chapter 17. (This is called the "ergodic decomposition" of the process: see Gray 2009.) We can think of this as first picking an ergodic process according to some fixed distribution, and then generating Y from that process. Time averages computed along any one trajectory thus converge according to Eq. 23.18. If we have only a single trajectory, it looks just like a stationary and ergodic process. It is thus common to assume that the data source is not only stationary but also ergodic. This only becomes a problem if we have multiple trajectories from the same source, each of which one may be converging to a different ergodic component.

Since this also applies to functions of blocks,

$$\frac{1}{n}\sum_{t=1}^{n} \mathbf{1}_{A,B}(X_t, X_{t+1}) \to \Pr\left(X_1 \in A, X_2 \in B\right) \tag{23.21}$$

and so on. If we can learn joint and marginal probabilities, and we remember how to divide, then we can learn conditional probabilities.

It turns out that pretty much any density estimation method which works for IID data will also work for getting the marginal and conditional distributions of time series (though, again, the effective sample size depends on how quickly dependence decays). So if we want to know $p(x_t)$, or $p(x_{t+1} \mid x_t)$, we can estimate it just as we learned how to do in Chapter 14. Just as in that chapter, much the same techniques apply whether x is discrete or continuous; for brevity, I'll speak as though x is continuous and $p(x_{t+1} \mid x_t)$ is a conditional pdf.

Now, the conditional distribution $p(x_{t+1} \mid x_t)$ always exists, and we can always estimate it. But why stop just one step back into the past? Why not look at $p(x_{t+1} \mid x_t, x_{t-1})$, or for that matter $p(x_{t+1} \mid x_{t-999:t})$? There are three reasons, in decreasing order of pragmatism.

- Estimating $p(x_{t+1} \mid x_{t-999:t})$ means estimating a thousand-and-one-dimensional distribution. The curse of dimensionality will crush us.
- Because of the decay of dependence, there shouldn't be much difference, much of the time, between $p(x_{t+1} \mid x_{t-999:t})$ and $p(x_{t+1} \mid x_{t-998:t})$, etc. Even if we could go very far back into the past, it shouldn't, usually, change our predictions very much.
- Sometimes, a finite, short block of the past completely screens off the remote past.

You will remember the Markov property from your previous probability classes:

$$X_{t+1} \perp\!\!\!\perp X_{1:t-1} \mid X_t \tag{23.22}$$

When the Markov property holds, there is simply no point in looking at $p(x_{t+1} \mid x_t, x_{t-1})$, because it's got to be just the same as $p(x_{t+1} \mid x_t)$. If the process isn't a simple Markov chain but has a higher-order Markov property,

$$X_{t+1} \perp\!\!\!\perp X_{1:t-k} \mid X_{t-k+1:t} \tag{23.23}$$

then we never have to condition on more than the last k steps to learn all that there is to know. The Markov property means that the current state screens off the future from the past.

It is *always* an option to model X_t as a Markov process, or a higher-order Markov process. If it isn't exactly Markov, if there's really some dependence between the past and the future even given the current state, then we're introducing some bias, but it can be small, and dominated by the reduced variance of not having to worry about higher-order dependencies.

23.3.1 Meaning of the Markov Property

The Markov property is a weakening both of being strictly IID and of being strictly deterministic.

That being Markov is weaker than being IID should be obvious: an IID sequence satisfies the Markov property, because everything is independent of everything else no matter what we condition on.

In a deterministic dynamical system, on the other hand, we have $X_{t+1} = g(X_t)$ for some fixed function g. Iterating this equation, the current state X_t fixes the whole future trajectory $X_{t+1}, X_{t+2}, \ldots$. In a Markov chain, we weaken this to $X_{t+1} = g(X_t, U_t)$, where the U_t are IID noise variables (which we can take to be uniform for simplicity). The current state of a Markov chain doesn't fix the exact future trajectory, but it does fix the *distribution* over trajectories.

The real meaning of the Markov property, then, is about information flow: the current state is the only channel through which the past can affect the future.

23.3.2 Estimating Markov Models

Once we believe that we're dealing with a Markov process, we really have only two things to have to estimate: the conditional distribution $p(x_{t+1} \mid x_t)$, and the initial distribution $p(x_1)$. Let's focus on the first, for reasons which will become apparent shortly. If X is continuous, then, as I said above, pretty much any method for estimating conditional distributions could be used. This is also true if X is discrete, but it's also possible to simplify matters.

Suppose that there are m states, so that we can collect all the probabilities $\Pr(X_{t+1} = j | X_t = i)$ in an $m \times m$ matrix $\mathbf{p}$ of **transition rates** or **transition probabilities**. Then the *conditional* likelihood of the time series, given the first observation x_1, is

$$\prod_{t=2}^{n} p_{X_{t-1} X_t} = \prod_{i,j} p_{ij}^{N_{ij}} \tag{23.24}$$

where I've introduced the (random) **transition counts**[5] N_{ij}, which tell us how many times the state i was followed by the state j. The log-likelihood is therefore

$$\sum_{i,j} N_{ij} \log p_{ij} \tag{23.25}$$

Before we can maximize this, we need to impose the constraint that each state is followed by *something*: for each i,

$$\sum_{j} p_{ij} = 1 \tag{23.26}$$

[5] It should be clear from the equation that if the process is Markovian, then any two sequences with the same transition counts are equally probable. It turns out that if the probability is the same for all sequences with equal transition counts, then the process must be Markovian, though the proof is intricate (Diaconis and Freedman, 1980).

We need to introduce m Lagrange multipliers to enforce these m constraints. Doing so, we get the very natural solution (Exercise 23.3):

$$\widehat{p}_{ij} = \frac{N_{ij}}{\sum_j N_{ij}} \tag{23.27}$$

Each time the process revisits state i, the next state is (by the Markov property) independent of the previous and subsequent visits, so using the law of large numbers, we have that $\widehat{p}_{ij} \to p_{ij}$, provided only that the state i is returned to infinitely often as n grows[6].

At this point, you may be thinking that this is very much like the maximum likelihood estimate of the multinomial distribution you'll have seen in baby stats. This is because it is *exactly* like that; each state i gets its own multinomial distribution for the next state, but those estimates can be done separately from each other. The maximum-likelihood estimate of $\mathbf{p}$ is, like the MLE for multinomial distributions, generally consistent and efficient, with a variance which can be found from the second derivative of the log-likelihood. Of course, if transition rates are not free to be adjusted independently, because they are all functions of more basic underlying parameters, we should estimate those parameters, which complicates the calculus a little.

Higher-order and variable-length Markov chains

In estimating a kth order Markov chain, $k > 1$, we still just need to estimate the transition rates, but the matrix of rates now has m^k rows (one per length-k history) and m columns. Each row must still sum to one, so the form of the solution remains unchanged; we just need to count transitions from length-k histories to the next observation.

Since m^k grows rapidly with k, it would be nice if we could get away from having to do that many estimations. It can happen that sometimes we don't need to keep track of *all* of the last k observations to get the next-observation distribution. For instance, in a second-order Markov process, it might happen that the history $X_t = 0, X_{t-1} = 0$ has the same predictive distribution as $X_t = 0, X_{t-1} = 1$, so we only need to estimate that distribution *once*, if we can realize this. Such approaches are known as "variable length Markov chains" or "context trees" (Bühlmann, 2000; Bühlmann and Wyner, 1999).

But what about that first observation?

By the Markov property, X_1 is irrelevant to the rest of the time series once we've seen X_2. The advantage of this is that the distribution of X_1 can be arbitrary, and we will still get consistent estimates of the transition rates. The disadvantage is that if, for some reason, we need to estimate the distribution of X_1, sometimes called the "starting" or "initial" distribution, we've got a problem, because we have only one observation!

[6] You might wonder how that last condition could fail, but consider a state which, once left, is never returned to from any other state. (Can you show that this implies a failure of ergodicity?)

If we see multiple independent realizations of the Markov chain and believe that they share a common starting distribution, we could use that to estimate X_1. On the other hand, if we believe the chain is stationary, we could use the transition rates to estimate the marginal distribution of all the X_t, as follows. Any distribution over the m states could be written as an m-dimensional vector, say q, with the constraints that $q_i \geq 0$, $\sum_i q_i = 1$. For q to match the marginal distribution over states in a stationary chain, the probability of arriving in any state q_i has to match the probability of starting there:

$$q_i = \sum_j q_j p_{ji} \tag{23.28}$$

In matrix form, this is

$$q = q\mathbf{p} \tag{23.29}$$

so q is an **invariant** distribution if it is a *left* eigenvector of $\mathbf{p}$ with eigenvalue one[7]. If we have an estimate $\widehat{p}$, finding its eigenvector(s) with eigenvalue 1 will then give an estimate of the invariant distribution, which (assuming stationarity) would be an estimate of the starting distribution[8]

23.4 Autoregressive Models

Instead of trying to estimate the whole conditional distribution of X_t, we can just look at its conditional expectation. This is a regression problem, but since we are regressing X_t on earlier values of the series, it's called an **autoregression**:

$$\mathbb{E}\left[X_t \mid X_{t-p:t-1} = x_1^p\right] = r(x_{1:p}) \tag{23.30}$$

If we think the process is Markov of order p, then of course there is no point in conditioning on more than p steps of the past when doing an autoregression. But even if we don't think the process is Markov, the same reasons which inclined us towards Markov approximations also make limited-order autoregressions attractive.

Since this is a regression problem, we can employ all the tools we know for regression analysis: linear models, kernel regression, spline smoothing, additive models, etc., mixtures of regressions, etc. Since we are regressing X_t on earlier values from the same series, it is useful to have tools for turning a time series into a regression-style design matrix (as in Figure 23.4); see Code Example 32.

[7] Conversely, all left eigenvectors of a transition matrix with eigenvalue one must have non-negative entries, and so must either be invariant distributions, or proportional to invariant distributions. This result is a non-trivial piece of linear algebra called the Frobenius-Perron (or Perron-Frobenius) theorem.

[8] You could even set up the problem of jointly maximizing the log-likelihood of the entire sequence, using the eigenvector of $\mathbf{p}$ as the distribution of X_1, but I don't recommend it. The eigenvector of $\mathbf{p}$ is a very nonlinear function of the entries in $\mathbf{p}$, so the maximization becomes a complicated numerical problem, and in the end it's only to get at the information about $\mathbf{p}$ contained in the *single* observation X_1. If X_1 is really very influential on $\widehat{\mathbf{p}}$, it's hard to imagine you've got enough data to be secure in all the other assumptions!

```
design.matrix.from.ts <- function(ts, order, right.older = TRUE) {
    n <- length(ts)
    x <- ts[(order + 1):n]
    for (lag in 1:order) {
        if (right.older) {
            x <- cbind(x, ts[(order + 1 - lag):(n - lag)])
        }
        else {
            x <- cbind(ts[(order + 1 - lag):(n - lag)], x)
        }
    }
    lag.names <- c("lag0", paste("lag", 1:order, sep = ""))
    if (right.older) {
        colnames(x) <- lag.names
    }
    else {
        colnames(x) <- rev(lag.names)
    }
    return(as.data.frame(x))
}
```

CODE EXAMPLE 32: *Example code for turning a time series into a design matrix, suitable for regression.*

```
aar <- function(ts, order) {
    stopifnot(require(mgcv))
    fit <- gam(as.formula(auto.formula(order)), data = design.matrix.from.ts(ts,
        order))
    return(fit)
}

auto.formula <- function(order) {
    inputs <- paste("s(lag", 1:order, ")", sep = "", collapse = "+")
    form <- paste("lag0 ~ ", inputs)
    return(form)
}
```

CODE EXAMPLE 33: *Fitting an additive autoregression of arbitrary order to a time series. See online for comments.*

Suppose $p = 1$. Then we essentially want to draw regression curves through plots like those in Figure 23.3. Figure 23.5 shows an example for the artificial series.

23.4.1 Autoregressions with Covariates

Nothing keeps us from adding a variable other than the past of X_t to the regression:

$$\mathbb{E}\left[X_{t+1} \mid X_{t-k+1:t}, Z\right] \tag{23.31}$$

or even another time series:

$$\mathbb{E}\left[X_{t+1} \mid X_{t-k+1:t}, Z_{t-l+1:t}\right] \tag{23.32}$$

These are perfectly well-defined conditional expectations, and quite estimable in principle. Of course, adding more variables to a regression means having to estimate more, so again the curse of dimensionality comes up, but our methods are very much the same as in the basic regression analyses.

23.4.2 Additive Autoregressions

As before, if we want some of the flexibility of non-parametric smoothing, without the curse of dimensionality, we can try to approximate the conditional expectation as an additive function:

$$\mathbb{E}\left[X_t \mid X_{t-p:t-1}\right] \approx \alpha_0 + \sum_{j=1}^{p} g_j(X_{t-j}) \tag{23.33}$$

My personal experience with applied projects is that additive autoregressions tend to work surprisingly well.

Example: The lynx

Let's try fitting an additive model for the lynx. Code Example 33 shows some code for doing this. (Most of the work is re-shaping the time series into a data frame, and then automatically generating the right formula for `gam`.) Let's try out $p = 2$.

```
lynx.aar2 <- aar(lynx, 2)
```

This inherits everything we can do with a GAM, so we can do things like plot the partial response functions (Figure 23.6), plot the fitted values against the actual (Figure 23.7), etc. To get a sense of how well it can actually extrapolate, Figure 23.8 re-fits the model to just the first 80 data points, and then predicts the remaining 34.

23.4.3 Linear Autoregression

When people talk about autoregressive models, they usually (alas) just mean linear autoregressions. There is almost never any justification in scientific theory for this preference, but we can always ask for the best linear approximation to the true autoregression, if only because it's fast to compute and fast to converge.

The analysis we did in Chapter 2 of how to find the optimal linear predictor carries over with no change whatsoever. If we want to predict X_t as a linear combination of the last k observations, $X_{t-1}, X_{t-2}, \ldots X_{t-p}$, then the ideal coefficients β are

$$\beta = \left(\mathbb{V}\left[X_{t-p:t-1}\right]\right)^{-1} \mathrm{Cov}\left[X_{t-p:t-1}, X_t\right] \tag{23.34}$$

where $\mathbb{V}[X_{t-p:t-1}]$ is the variance-covariance matrix of $(X_{t-1}, \ldots X_{t-p})$ and similarly $\mathrm{Cov}[X_{t-p:t-1}, X_t]$ is a vector of covariances. Assuming stationarity, $\mathbb{V}[X_t]$ is constant in t, and so the common factor of the over-all variance goes away, and β could be written entirely in terms of the correlation function ρ. Stationarity also lets us estimate these covariances, by taking time-averages.

A huge amount of effort is given over to using linear AR models, which in my opinion is out of all proportion to their utility — but very reflective of what was computationally feasible up to about 1980. My experience is that results like Figure 23.9 is pretty typical.

23.4.3.1 "Unit Roots" and Stationary Solutions

Suppose we really believed a first-order linear autoregression,

$$X_{t+1} = \alpha + \beta X_t + \epsilon_t \tag{23.35}$$

with ϵ_t some IID noise sequence. Let's suppose that the mean is zero for simplicity, so $\alpha = 0$. Then

$$X_{t+2} = \beta^2 X_t + \beta\epsilon_t + \epsilon_{t+1} \tag{23.36}$$

$$X_{t+3} = \beta^3 X_t + \beta^2\epsilon_t + \beta\epsilon_{t+1} + \epsilon_{t+2}\ , \tag{23.37}$$

etc. If this is going to be stationary, it'd better be the case that what happened at time t doesn't go on to dominate what happens at all later times, but clearly that will happen if $|\beta| > 1$, whereas if $|\beta| < 1$, eventually all memory of X_t (and ϵ_t) fades away. The linear AR(1) model in fact can only produce stationary distributions when $|\beta| < 1$.

For higher-order linear AR models, with parameters $\beta_1, \beta_2, \ldots \beta_p$, the corresponding condition is that all the roots of the polynomial

$$\sum_{j=1}^{p} \beta_j z^j - 1 \tag{23.38}$$

must be outside the unit circle. When this fails, when there is a "unit root", the linear AR model cannot generate a stationary process[9].

There is a fairly elaborate machinery for testing for unit roots, which is sometimes also used to test for non-stationarity. It is not clear how much this really matters. A non-stationary but truly linear AR model can certainly be estimated[10]; a linear AR model can be non-stationary even if it has no unit roots[11]; and if the linear model is just an approximation to a non-linear one, the unit-root criterion doesn't apply to the true model anyway.

See §23.6.1 for an alternative way of checking stationarity, which presumes no particular parametric form.

[9] The same argument applies to ARMA models (§23.9.3.2) more generally.

[10] Because the correlation structure stays the same, even as the means and variances can change. Consider $X_t = X_{t-1} + \epsilon_t$, with ϵ_t IID.

[11] Start it with X_1 very far from the expected value.

23.4.4 Conditional Variance

Having estimated the conditional expectation, we can estimate the conditional variance $\mathbb{V}[X_t \mid X_{t-p:t-1}]$ just as we estimated other conditional variances, in Chapter 10.

Example: lynx

The lynx series seems ripe for fitting conditional variance functions — presumably when there are a few thousand lynxes, the noise is going to be larger than when there are only a few hundred.

```
sq.res <- residuals(lynx.aar2)^2
lynx.condvar1 <- gam(sq.res ~ s(lynx[-(1:2)]))
lynx.condvar2 <- gam(sq.res ~ s(lag1) + s(lag2), data = design.matrix.from.ts(lynx,
    2))
```

I have fit two different models for the conditional variance here, just because. Figure 23.10 shows the data, and the predictions of the second-order additive AR model, but with just the standard deviation bands corresponding to the first of these two models; you can try making the analogous plot for `lynx.condvar2`.

23.4.5 Regression with Correlated Noise; Generalized Least Squares

Suppose we have an old-fashioned regression problem

$$Y_t = \mu(X_t) + \epsilon_t \tag{23.39}$$

only now the noise terms ϵ_t are themselves a dependent time series. Ignoring this dependence, and trying to estimate μ by minimizing the mean squared error, is very much like ignoring heteroskedasticity. (In fact, heteroskedastic ϵ_t are a special case.) What we saw in Chapter 10 is that ignoring heteroskedasticity doesn't lead to bias, but it does mess up our understanding of the uncertainty of our estimates, and is generally inefficient. The solution was to weight observations, with weights inversely proportional to the variance of the noise.

With correlated noise, we do something very similar. Suppose we knew the covariance function $\gamma(h)$ of the noise. From this , we could construct the variance-covariance matrix $\mathbf{\Gamma}$ of the ϵ_t (since $\Gamma_{ij} = \gamma(i-j)$, of course).

We can use this as follows. Say that our guess about the regression function is m. Stacking $y_1, y_2, \ldots y_n$ into a matrix $\mathbf{y}$ as usual in regression, and likewise creating $\mathbf{m}(x)$, the Gauss-Markov theorem (§10.2.2.1) tells us that the most efficient estimate is the solution to the **generalized least squares** problem,

$$\widehat{m}_{GLS} = \operatorname*{argmin}_m \frac{1}{n}(\mathbf{y} - \mathbf{m}(\mathbf{x}))^T \Gamma^{-1} (\mathbf{y} - \mathbf{m}(\mathbf{x})) \tag{23.40}$$

as opposed to just minimizing the mean-squared error,

$$\widehat{m}_{OLS} = \operatorname*{argmin}_m \frac{1}{n}(\mathbf{y} - \mathbf{m}(\mathbf{x}))^T (\mathbf{y} - \mathbf{m}(\mathbf{x})) \tag{23.41}$$

Multiplying by the inverse of Γ appropriately discounts for observations which are very noisy, *and* discounts for correlations between observations introduced by the noise.[12]

This raises the question of how to get $\gamma(h)$ in the first place. If we knew the true regression function μ, we could use the covariance of $Y_t - \mu(X_t)$ across different t. Since we don't know μ, but have only an estimate $\hat{m}$, we can try alternating between using a guess at γ to estimate $\hat{m}$, and using $\hat{m}$ to improve our guess at γ. We used this sort of iterative approximation for weighted least squares, and it can work here, too.

23.5 Bootstrapping Time Series

The big picture of bootstrapping doesn't change: simulate a distribution which is close to the true one, repeat our estimate (or test or whatever) on the simulation, and then look at the distribution of this statistic over many simulations. The catch is that the surrogate data from the simulation has to have the same sort of dependence as the original time series. This means that simple resampling is just wrong (unless the data are independent), and our simulations will have to be more complicated.

23.5.1 Parametric or Model-Based Bootstrap

Conceptually, the simplest situation is when we fit a full, generative model — something which we could step through to generate a new time series. If we are confident in the model specification, then we can bootstrap by, in fact, simulating from the fitted model. This is the parametric bootstrap we saw in Chapter 6.

23.5.2 Block Bootstraps

Simple resampling won't work, because it destroys the dependence between successive values in the time series. There is, however, a clever trick due to Künsch (1989) which does work, and is almost as simple. Take the full time series $x_{1:n}$ and divide it up into overlapping **blocks** of length k, i.e., $x_{1:k}$, $x_{2:k+1}$ and so on down to $x_{n-k+1:n}$. Now draw $m = n/k$ of these *blocks* with replacement[13], and set them down in order. Call the new time series $\tilde{x}_{1:n}$.

Within each block, we have preserved *all* of the dependence between observations. It's true that successive observations are now completely independent, which generally wasn't true of the original data, so we're introducing some inaccuracy, but we're certainly coming closer than just resampling individual observations (which would be $k = 1$). Moreover, we can make this inaccuracy smaller and smaller by letting k grow as n grows. One can show[14] that the optimal

[12] If you want to use a linear model for m, this can be carried through to an explicit modification of the usual ordinary-least-squares estimate — Exercise 23.4.

[13] If n/k isn't a whole number, round.

[14] I.e., I will not show — see Lahiri (2003)..

```
rblockboot <- function(ts, block.length, len.out = length(ts)) {
    the.blocks <- as.matrix(design.matrix.from.ts(ts, block.length - 1, right.older = FALSE))
    blocks.in.ts <- nrow(the.blocks)
    stopifnot(blocks.in.ts == length(ts) - block.length + 1)
    blocks.needed <- ceiling(len.out/block.length)
    picked.blocks <- sample(1:blocks.in.ts, size = blocks.needed, replace = TRUE)
    x <- the.blocks[picked.blocks, ]
    x.vec <- as.vector(t(x))
    return(x.vec[1:len.out])
}
```

CODE EXAMPLE 34: *The basic block bootstrap for univariate time series. See Exercise 23.6 for variants and extensions.*

$k = O(n^{1/3})$; this gives a growing number ($O(n^{2/3})$) of increasingly long blocks, capturing more and more of the dependence. (We will consider how exactly to pick k [[below]].)

The block bootstrap scheme is extremely clever, and has led to a great many variants. Three in particular are worth mentioning.

1. In the **circular block bootstrap** (or **circular bootstrap**), we "wrap the time series around a circle", so that it goes $x_1, x_2, \ldots x_{n_1}, x_n, x_1, x_2, \ldots$. We then sample the n blocks of length k this gives us, rather than the merely $n - k$ blocks of the simple block bootstrap. This makes better use of the information we have about dependence on distances $< k$.
2. In the **block-of-blocks bootstrap**, we first divide the series into blocks of length k_2, and then subdivide each of those into sub-blocks of length $k_1 < k_2$. To generate a new series, we sample blocks with replacement, and then sample sub-blocks within each block with replacement. This gives a somewhat better idea of longer-range dependence, though we have to pick two block-lengths.
3. In the **stationary bootstrap**, the length of each block is random, chosen from a geometric distribution of mean k. Once we have chosen a sequence of block lengths, we sample the appropriate blocks with replacement. The reason for doing this is that the ordinary block bootstrap doesn't quite give us a stationary time series, since the distribution gets funny around the boundaries between blocks. (The distribution of $X_{k-1:k}$ is not the same as the distribution of $X_{k:k+1}$, as stationarity would require.) Averaging over the random choices of block lengths, the stationary bootstrap does. It tends to be slightly slower to converge that the block or circular bootstrap, but there are some applications where the surrogate data really needs to be strictly stationary.

23.5.3 Sieve Bootstrap

A compromise between model-based and resampling bootstraps is to use a **sieve bootstrap**. This also simulates from models, but we don't really believe in them; rather, we just want them to be reasonable easy to fit and simulate, yet flexible enough that they can capture a wide range of processes if we just give them

enough capacity. We then (slowly) let them get more complicated as we get more data[15]. One popular choice is to use linear AR(p) models, and let p grow with n — but there is nothing special about linear AR models, other than that they are very easy to fit and simulate from. Additive autoregressive models, for instance, would often work at least as well.

23.6 Cross-Validation

There are actually multiple ways to do cross-validation for time series.

The most straight-forward applies to auto-regressive models. Since we have, at least implicitly, converted the time series to a design matrix (as on p. 527), we can "hold out" rows from that design matrix at random as our testing set. Concretely, this means that we don't try to predict some time points $t \in T_{test}$ when estimating the model. (The set T_{test} is randomly chosen, and then averaged-over, in the usual way.) Those held-out time-points t are then what we try to predict during testing, using the previously-estimated model and the predecessor points $t-1, t-2, \ldots t-p$.

Notice that even if t is a held-out time-point, we are still likely to use $X(t)$ to predict (say) $X(t+1)$, since it is unlikely that both t and $t+1$ are in the same hold-out set. Nothing like this happened when doing cross-validation for IID data — every observed value was either in the training set or the testing set, not awkwardly straddling the border between them. If we want something like this under serial dependence, the natural approach is to remove a **buffer** around each testing point. The points in this buffer are *not* used during training, but just to provide input values when trying to predict the points in the test set.

23.6.1 Testing Stationarity by Cross-Prediction

[[After Schreiber (1997) and the version in Kantz and Schreiber (2004)]]

23.7 Trends and De-Trending

The sad fact is that a lot of important time series are not even approximately stationary. For instance, Figure 23.13 shows US national income per person (adjusted for inflation) over the period from 1952 (when the data series begins) until the last time I re-ran the code for this book. It is *possible* that this is sample from a stationary process. But in that case, the correlation time is evidently much longer than 50 years, on the order of centuries, and so the theoretical stationarity is irrelevant for anyone but a very ambitious quantitative historian — living in our distant future.

It makes more sense to treat data like this as a non-stationary time series.

[15] This is where the metaphor of the "sieve" comes in: the idea is that the mesh of the sieve gets finer and finer, catching more and more subtle features of the data.

The conventional approach is try decomposing such time series into a persistent **trend**, and stationary **fluctuations** (or **deviations**) around the trend,

$$Y_t = X_t + Z_t \tag{23.42}$$
$$\text{series} = \text{fluctuations} + \text{trend}$$

Since we could add or subtract a constant to each X_t without changing whether they are stationary, we'll stipulate that $\mathbb{E}[X_t] = 0$, so $\mathbb{E}[Y_t] = \mathbb{E}[Z_t]$. (In other situations, the decomposition might be multiplicative instead of additive, etc.) How might we find such a decomposition?

If we have multiple independent realizations $Y_{i,t}$ of the same process, say m of them, and they all have the same trend Z_t, then we can try to find the common trend by averaging the time series:

$$Z_t = \mathbb{E}[Y_{\cdot,t}] \approx \sum_{i=1}^{m} Y_{i,t} \tag{23.43}$$

Multiple time series with the same trend do exist, especially in the experimental sciences. $Y_{i,t}$ might be the measurement of some chemical in a reactor at time t in the ith repetition of the experiment, and then it would make sense to average the $Y_{i,t}$ to get the common Z_t trend, the average trajectory of the chemical concentration. One can tell similar stories about experiments in biology or even psychology, though those are complicated by the tendency of animals to get tired and to learn[16].

For better or for worse, however, we have only one realization of the post-WWII US economy, so we can't average multiple runs of the experiment together. If we have a theoretical model of the trend, we can try to fit that model. For instance, some (simple) models of economic growth predict that series like the one in Figure 23.13 should, on average, grow at a steady exponential rate[17]. We could then estimate Z_t by fitting a model to Y_t of the form $\beta_0 e^{\beta t}$, or even by doing a linear regression of $\log Y_t$ on t. The fluctuations X_t are then taken to be the residuals of this model.

If we only have one time series (no replicates), and we don't have a good theory which tells us what the trend should be, we fall back on curve fitting. In other words, we regress Y_t on t, call the fitted values Z_t, and call the residuals X_t. This is frankly rests more on hope than on theorems. The hope is that the characteristic time-scale for the fluctuations X_t (say, their correlation time τ) is short compared to the characteristic time-scale for the trend Z_t[18]. Then if we average Y_t over a

[16] Even if we do have multiple independent experimental runs, it is very important to get them aligned in time, so that $Y_{i,t}$ and $Y_{j,t}$ refer to the *same* point in time relative to the start of the experiment; otherwise, averaging them is just mush. It can also be important to ensure that the initial state, before the experiment, is the same for every run. Chu *et al.* (2003) explains how the later problem can lead to complications in studying gene regulation.

[17] This is not *quite* what is claimed by Solow (1970), which you should read anyway if economic growth interests you at all.

[18] I am being deliberately vague about what "the characteristic time scale of Z_t" means. Intuitively, it's the amount of time required for Z_t to change substantially. You might think of it as something

band-width which is large compared to τ, but small compared to the scale of Z_t, we should get something which is mostly Z_t — there won't be too much bias from averaging, and the fluctuations should mostly cancel out.

Once we have the fluctuations, and are reasonably satisfied that they're stationary, we can model them like any other stationary time series. Of course, to actually make predictions, we need to extrapolate the trend, which is a harder business.

23.7.1 Forecasting Trends

The problem with making predictions when there is a substantial trend is that it is usually hard to know how to continue or extrapolate the trend beyond the last data point. If we are in the situation where we have multiple runs of the same process, we can at least extrapolate up to the limits of the different runs. If we have an actual model which tells us that the trend should follow a certain functional form, and we've estimated that model, we can use it to extrapolate. But if we have found the trend purely through curve-fitting, we have a problem.

Suppose that we've found the trend by spline smoothing, as in Figure 23.16. The fitted spline model will cheerfully make predictions for the what the trend of GDP per capita will be in, say, 2252, far outside the data. This will be a simple linear extrapolation, because splines are always linear outside the data range (Chapter 7, p. 170). This is just because of the way splines are set up, not because linear extrapolation is such a good idea. Had we used kernel regression, or any of many other ways of fitting the curve, we'd get different extrapolations. People in 2252 could look back and see whether the spline had fit well, or some other curve would have done better. (But why would they want to?) Right now, if *all* we have is curve-fitting, we are in a dubious position even as regards next year, never mind 2252[19]

23.7.2 Seasonal Components

Sometimes we know that time series contain components which repeat, pretty exactly, over regular periods. These are called **seasonal** components, after the obvious example of trends which cycle each year with the season. But they could cycle over months, weeks, days, etc.

like $n^{-1}\sum_{t=1}^{n-1} 1/|Z_{t+1} - Z_t|$, if you promise not to treat that too seriously. Trying to get an exact statement of what's involved in identifying trends requires being very precise, and getting into topics at the intersection of statistics and functional analysis which are beyond the scope of this class.

[19] Yet again, we hit a basic philosophical obstacle, which is the problem of induction. We have so far evaded it, by assuming that we're dealing with IID or a stationary probability distribution; these assumptions let us *deductively* extrapolate from past data to future observations, with more or less confidence. (For more on this line of thought, see Hacking (2001); Spanos (2011); Gelman and Shalizi (2013).) If we assume a certain form or model for the trend, then again we can deduce future behavior on that basis. But if we have neither probabilistic nor mechanistic assumptions, we are, to use a technical term, stuck with induction. Whether there is some principle which might help — perhaps a form of Occam's Razor (Kelly, 2007)? — is a nice question.

The decomposition of the process is thus

$$Y_t = X_t + Z_t + S_t \tag{23.44}$$

where X_t handles the stationary fluctuations, Z_t the long-term trends, and S_t the repeating seasonal component.

If $Z_t = 0$, or equivalently if we have a good estimate of it and can subtract it out, we can find S_t by averaging over multiple cycles of the seasonal trend. Suppose that we know the period of the cycle is T, and we can observe $m = n/T$ full cycles. Then

$$S_t \approx \frac{1}{m}\sum_{j=0}^{m-1} Y_{t+jT} \tag{23.45}$$

This works because, with Z_t out of the picture, $Y_t = X_t + S_t$, and S_t is periodic, $S_t = S_{t+T}$. Averaging over multiple cycles, the stationary fluctuations tend to cancel out (by the ergodic theorem), but the seasonal component does not.

For this trick to work, we need to know the period. If the true $T = 355$, but we use $T = 365$ without thinking[20], we'll get mush.

We also need to know the over-all trend. Of course, if there are seasonal components, we really ought to subtract them out before trying to find Z_t. So we have yet another vicious cycle, or, more optimistically, another case for iterative approximation.

23.7.3 Detrending by Differencing

Suppose that Y_t has a linear time trend:

$$Y_t = \beta_0 + \beta t + X_t \tag{23.46}$$

with X_t stationary. Then if we take the difference between successive values of Y_t, the trend goes away:

$$Y_t - Y_{t-1} = \beta + X_t - X_{t-1} \tag{23.47}$$

Since X_t is stationary, $\beta + X_t - X_{t-1}$ is also stationary. Taking differences has removed the trend.

Differencing will not only get rid of linear time trends. Suppose that

$$Z_t = Z_{t-1} + \epsilon_t \tag{23.48}$$

where the "innovations" or "shocks" ϵ_t are IID, and that

$$Y_t = Z_t + X_t \tag{23.49}$$

with X_t stationary, and independent of the ϵ_t. It is easy to check that (i) Z_t is not stationary (Exercise 23.5), but that (ii) the first difference

$$Y_t - Y_{t-1} = \epsilon_t + X_t - X_{t-1} \tag{23.50}$$

[20] Can you come up with an example of a time series where the periodicity *should* be 355 days?

is stationary. So differencing can get rid of trends which are built out of the summation of *persistent* random shocks.

Differencing gives us another way of making a time series stationary: instead of trying to model the time trend, take the difference between successive values, and see if that is stationary. (The `diff()` function in R does this; see Figure 23.18.) If such "first differences" don't look stationary, take differences among differences, third differences, etc., until you have something satisfying.

Differencing is like taking the discrete version of a derivative. Repeated differencing will eventually get rid of trends if they correspond to curves (e.g., polynomials) with only finitely many non-zero derivatives. It fails for trends which aren't like that, like exponentials or sinusoids, though you can hope that eventually the higher differences are small enough that they don't matter much.

Notice that now we can continue to the trend (a little): once we predict $Y_{t+1} - Y_t$, we add it on to Y_t (which we observed) to get Y_{t+1}.

23.7.4 Cautions with Detrending

The fact that I've explained multiple different ways of detrending non-stationary time series may have made you uneasy: how are you to know which one to use? My unhelpful answer is "it depends", namely, on what you think is a plausible about the trend and the fluctuations around it. (E.g., if you think the trend is linear, then differencing should work.) My *advice* is to try several different ways of detrending your data, and to examine them very carefully if they give substantially different results.

Finally, it is worth considering how much damage you might do by de-trending if the process really is stationary. E.g., if the original series is really uncorrelated, differencing will *create* correlations — see Exercise 23.7, and §23.9.2 on the Yule-Slutsky effect.

23.7.5 Bootstrapping with Trends

All the bootstraps discussed in §23.5 work primarily for stationary time series. (Parametric bootstraps are an exception, since we *could* include trends in the model.) If we have done extensive de-trending, the reasonable thing to do is to use a bootstrap to generate a series of fluctuations, add it to the estimated trend, and then repeat the *whole* analysis on the new, non-stationary surrogate series, including the de-trending. This works on the same sort of principle as resampling residuals in regressions (§6.4, especially 6.4.3).

23.8 Breaks in Time Series

Figure 23.19 shows the employment to population ratio[21] for the US since 1990. There are fairly periodic oscillations — it's not seasonally adjusted — but it seems to be fluctuating within a not-too-wide band, and then 2008 happens, and the Lesser Depression begins.

What should we, as time series analysts, do with something like this? It goes against intuition to say that this sort of abrupt and dramatic break is all part of a single stationary process, but by this point I hope you are all thoroughly suspicious of that sort of intuition. The two big routes to dealing with series which look like this are (1) to treat them as stationary, never mind our gut, or (2) to give up on global stationarity, to say that sometimes things just change abruptly.

23.8.1 Long Memory Series

The simplest option for dealing with series that look like Figure 23.19 is to say that they are *really* stationary time series, except that the decay of dependence is very slow — that the time series has a **long memory**. Formally, a long-memory time series is one where the covariance function $\gamma(h) = O(h^{-\alpha})$ for some $\alpha > 0$. If α is big enough, $\sum_{h=0}^{\infty} |\gamma(h)|$ is still finite — but the slow decay of $\gamma(h)$ means that the sum, and so the correlation time, is quite large. A large correlation time means that we need to wait a very long time before any one trajectory becomes representative of the whole system — in this case, perhaps, several centuries.[22]

23.8.2 Change Points and Structural Breaks

We could of course give up on the idea that all the data come from a single stationary process. The most popular alternative is the idea of a **change point** or **structural break**. Up to some time, call it t_b, the process followed one stationary process. After this change point, it follows a different stationary process, perhaps bearing no relationship at all to what went before.

If we think we're dealing with a change point, the natural questions are, When did it change?, and What does the process look like after the change? Before we plunge in to those questions, however, let's look at the contrast between change points and long memory.

23.8.2.1 Change Points and Long Memory

Suppose that the change-point manifests itself by a shift in the expectation value of X_t, say from μ_1 before the change to μ_2 after. The global mean of the time series $n^{-1}\sum_t X_t$ is the somewhere between μ_1 and μ_2. If h is not too large, then

[21] That is, the ratio of the total number of employed people to all people. This is not one minus the unemployment rate, because the denominator in the unemployment rate excludes those who wouldn't be looking for paid work anyway, such as retirees.

[22] See also §23.9.3.3 on "regime switching" models.

for most t, X_t and X_{t+h} will be on the same side of the change point. If they are both before, then X_t and X_{t+h} will both be somewhere around μ_1, and if both times are after the point, both values will be around μ_2. Therefore, it will tend to be the case that either both X_t and X_{t+h} are above the global mean, or both of them are below it — and so they're correlated. This argument applies even if the X_t are really all independent, as in Figure 23.20.

This phenomenon makes it *very* hard to distinguish empirically between time series which have change points and those which have a slow decay of dependence.

23.8.3 Change Point Detection

It is often reasonable to set aside such scruples, assume there are change points, and try to find them. A large number of methods have been developed for this purpose, often under very strong parametric restrictions — say that $X_t \sim_{IID} \mathcal{N}(\mu_1, \sigma^2)$ when $t < t_b$, and $X_t \sim_{IID} \mathcal{N}(\mu_2, \sigma^2)$ when $t \geq t_b$. Many of these have the flavor of looking for "runs" of values which are cumulatively very unlikely — for instance, we might look for a long run of values which are far from μ_1 and on the same side of it. Other procedures boil down to "will dividing this time series here, and letting the parameters change, work better?" Along those lines, it is natural to try to use cross-validation, and Arlot and Celisse (2011) propose a segmentation algorithm on exactly such a basis.

23.9 Time Series with Latent Variables

There are many time-series problems where we want to use a model where one or more of the time series are latent. The best reason for to do this is, of course, that there really is a latent process related to the observables, or at least that we think their might be one. It can also make models more interpretable, if the dynamics of the latent process are somehow simpler than those of the observables. It is sometimes done just to increase the capacity of the model, and soak up some mis-specification error; this is not a good practice, but it is common.

While there an almost infinite variety of time series models with latent variables are possible, people generally work with schemes that share some common features. The most important of these is that these models sum up everything we need from the past of the process into one (possibly-multidimensional variable), the **state**. The state evolves according to a Markov process, and the current state completely fixes the distribution of all present and future observables. We generally do not get to directly observe the state, so it is latent, but it is supposed to drive, or at least summarize, everything which we do observe. That is, the state at time t, S_t, obeys the Markov property, so $S_{t+1:\infty} \perp\!\!\!\perp S_{-\infty:t-1} | S_t$, and screens off the future of the observables from the past of the whole process, $X_{t:\infty} \perp\!\!\!\perp X_{-\infty:t-1}, S_{-\infty:t-1} | S_t$.

Models of this sort are called "state-space models", or "partially-observable Markov models". There are, however, (at least) two kinds of models which meet

all these requirements, most easily distinguished through their graphical models. Figures 23.21 and Figure 23.22 show the alternatives. In both cases, S_t has as its parent S_{t-1} and as its child S_{t+1}, and the S's form a Markov process. In both cases, S_t is the sole parent of X_t, so that conditioning on S_t makes X_t independent of all that came before. Where they differ is in X_t's children. In Figure 23.21, X_t has no children. What we observe at time t might *tell us* about how the state evolves, but it doesn't *change* how the state evolves. In Figure 23.22, however, X_t and S_t are the two parents of S_{t+1}, which indeed is usually taken to be a *deterministic* function of its parents. In this alternative, what we observe at time t directly affects the future state. Former alternative (Figure 23.21) has come to be called a **hidden Markov model** (HMMs), while the latter (Figure 23.22) is known as a **chain with complete connections** (CCCs). In both cases, you can verify that the X's do *not* form a Markov process.

While these are both logically possible and mathematically interesting, statistical practice has favored HMMs over CCCs, to the point where "state space model", when used in time series analysis or econometrics, almost always means an HMM. However, CCCs have important uses, and there doesn't seem to be any really deep reason why they are not used more in statistics.

[["Parameter-driven" vs. "observation-driven"]]

Regardless of which kind of state-space model you want to use, there are four basic problems for them:

- Simulation: how to generate new time series from a fully-parameterized model?
- Prediction: how, given a sequence of observations $X_{1:t}$ and a parameterized model, do we come up with a guess for X_{t+1}?
- State estimation: how, given a fully-parameterized model and the time series of X's, to find estimates of the latent states S?
- Inference: how do we estimate the parameters of the model (or fit it nonparametrically), or test guesses about the parameters?

Simulation is straightforward, at least in outline. Our general strategy for simulation (§5.2.1) tells us to start with generating S_1, then generate X_1 from the conditional distribution of $X_1|S_1$, etc., etc. In short, we write out the DAG, and we generate each variable by conditioning on its parents. There can be clever tricks for doing simulation faster in special situations, but this is the core idea.

Prediction is also straightforward in principle. We would, ideally, like the distribution $p(x_{t+1}|x_{1:t})$, since we can calculate any other prediction (say, a 90%

prediction interval) from this. But this is

$$p(x_{t+1}|x_{1:t}) = \sum_{s_{1:t+1}} p(x_{t+1}|s_{1:t+1}, x_{1:t})p(s_{1:t+1}|x_{1:t}) \tag{23.51}$$

$$= \sum_{s_{1:t+1}} p(x_{t+1}|s_{t+1})p(s_{t+1}|x_{1:t}, s_{1:t})p(s_{1:t}|x_{1:t}) \tag{23.52}$$

$$= \sum_{s_{1:t+1}} p(x_{t+1}|s_{t+1})p(s_{t+1}|x_t, s_t)p(s_{1:t}|x_{1:t}) \tag{23.53}$$

$$= \sum_{s_{t:t+1}} p(x_{t+1}|s_{t+1})p(s_{t+1}|x_t, s_t)p(s_t|x_{1:t}) \tag{23.54}$$

(For an HMM, $p(s_{t+1}|x_t, s_t)$ itself simplifies to $p(s_{t+1}|s_t)$.) Thus the key part of prediction turns out to be doing state estimation, specifically finding the distribution $p(s_t|x_{1:t})$.

State estimation is also important for inference, since the observable likelihood, $p(x_{1:n})$, is a product of predictive distributions:

$$p(x_{1:n}) = p(x_1)\prod_{t=2}^{n} p(x_t|x_{1:t-1}) \tag{23.55}$$

Though, it must be said, there are ways of doing inference which don't rely on the likelihood, some of which can side-step state estimation.

The bulk of my treatment of time series models with latent variables will, therefore, be devoted to state estimation, since it's crucial to the other statistical problems with these models. Before plunging in to these details, however, it is instructive to first consider the simplest, and historically oldest, time series model with latent variables, which is just "the time series is a moving average of random noise" (§23.9.1), and then some more complex examples (§23.9.3), before diving in (§23.9.4).

23.9.1 Moving Averages and Apparent Cycles

The basic equation for a moving average (MA) model of order q, or MA(q) is

$$X_t = Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i} \tag{23.56}$$

with the Z_t being IID noise terms. That is, what we observe is a weighted average[23] of the $q+1$ most recent noise variables.

Figure 23.23 shows the graphical model for an MA(1) model. It's evident from it that $X_t \not\perp\!\!\!\perp X_{t-1}$, but $X_t \perp\!\!\!\perp X_{t-k}$, $k > 1$ — observables are only dependent on each other through the hidden noise variables, and X_t and X_{t-k} have no common parents. In general, in an MA(q), $X_t \perp\!\!\!\perp X_{t-k}$ when $k > q$.

[23] The right-hand side would look more like a weighted *average* if we wrote it $X_t = \frac{Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i}}{1+\sum_{i=1}^{q} \theta_i}$, but since the Z_t are latent we could just re-scale each of them by the denominator. (Likewise, we can always impose weight 1 on the most recent Z.)

Suppose that we try to predict X_t from its past values. We condition X_t on X_{t-1}, and ask whether there is still more information to be had about X_t from X_{t-2}. This is asking whether X_t and X_{t-2} are dependent, given X_{t-1}. The answer is clearly yes from Figure 23.23: there is one path linking X_t to X_{t-2}, and X_{t-1} is a collider on that path, so conditioning on it activates the path.

Why does X_{t-2} give us information about X_t, conditional on X_{t-1}? To determine X_t, we'd need to know Z_t and Z_{t-1}. Since X_{t-1} is a child of Z_{t-1} and Z_{t-2}, knowing X_{t-1} tells us something about Z_{t-1}, but we learn even more from *also* knowing X_{t-2}.

Undaunted, we try conditioning X_t on $X_{t-2:t-1}$. Is $X_t \perp\!\!\!\perp X_{t-3} | X_{t-2:t-1}$? Clearly not. There is again only a single path, which goes over two colliders — and we condition on both of them, activating the path. Knowing X_{t-3} would tell us more about Z_{t-3}, and that, with X_{t-2}, tells us more about Z_{t-2}, which, together with X_{t-1}, helps us pin down Z_{t-1} even better. The chain of inferences is getting longer and longer, but it's not breaking, and it's evident that it will never break, no matter how many steps back into the past we condition.

To sum up, an MA(1) process, and by extension any MA(q), is not Markov, no matter what order of Markov chain we consider. Nonetheless, all of the dependence of future on the past is carried by a simple, low-dimensional state variable, (Z_{t-1}, Z_t). Conditional on that, X_t is independent of all other X_s's[24].

23.9.2 Yule-Slutsky

Applying a moving average to independent noise creates a process with complicated dependence. This fact was noticed independently by two pioneers of time series analysis, G. Udny Yule and E. E. Slutsky. It is therefore known as the **Yule-Slutsky effect**. But Yule and Slutsky gave very different interpretations to it — both are valid in their own circumstances, but the contrast is instructive.

23.9.2.1 Slutsky

Slutsky was primarily interested under the fluctuations of the economy — in the business cycle. The way he thought of a moving average process was that the economy is (under capitalism) continually subjected to random, unpredictable shocks, but it *takes time* for the economy to respond to them, for them to work their way through the machinery (as it were). The coefficients θ represent how the economy responds over time to any given shock. That this leads to fluctuations with a characteristic amplitude and (nearly) duration was a feature, not a bug — it was how Slutsky proposed to *explain* the business cycle[25]. It is not at all clear that any subsequent theory of the business cycle has any more *predictive* power (cf. Figure **??**).

[24] Because Z_{t-1} and Z_t are the only parents of X_t, which has no descendants.

[25] The USSR in the 1920s being what it was, Slutsky had to do some fast talking to try to reconcile this with Marxism. (In particular, depicting capitalism as *stationary*, rather than inevitably self-destructive, was not very politically correct.) He was lucky to be allowed to escape into pure probability theory (Klein, 1997, pp. 276–279); many of his colleagues were not so lucky.

23.9.2.2 Yule

Moving averages are of course a very common way of smoothing time series. We can think of them as being rather like kernel smoothing, but with a one-sided kernel. That is, we start with our original data Z_t, and then average it together locally to get a smoother series X_t, with some of the noise removed. What Yule recognized is that doing this will, all by itself, create correlations among the X_t (cf. Chapter 4), and complicated predictive relationships. Indeed, even if the Z_t are all independent of each other, the X_t will be correlated, and will have non-zero linear regression coefficients (or other regression functions, if you use them). Part of what we infer on the X_t is then just the effects of our smoothing.

This Yule effect is very basic, and very easy to understand as soon as one sees Figure 23.23, but it continues to trip up researchers in a wide range of applied fields[26] Don't be like that.

23.9.3 Examples of State-Space Models

23.9.3.1 General Gaussian-Linear State Space Model

The classical example of a hidden Markov model has a state variable S evolving linearly, subject to noise,

$$S_t = \mathbf{a}S_{t-1} + \eta_t \tag{23.57}$$

but what we observe being a noisy, linear function of the hidden state,

$$X_t = \mathbf{b}X_t + \epsilon_t \tag{23.58}$$

It is often assumed that η_t and ϵ_t are each IID series (generally with different distributions), and independent of each other. This is the general linear state space model. If one further assumes that both the dynamical noise η and the observation noise ϵ are Gaussian, then one gets a general Gaussian-linear state space model. When the parameters are known, the Kalman filter provides an exact, closed-form formula for estimating the latent state S_t and updating it as new observations are made (Kalman, 1960; Kalman and Bucy, 1961). This in turn forms a component of estimating the parameters by maximum likelihood. Because this model is very extensively treated elsewhere, we will say almost nothing more about it, but refer the interested reader to Durbin and Koopman (2001) or Fraser (2008).

23.9.3.2 Autoregressive-Moving Average (ARMA) Models

An important result in the theory of stochastic processes says that basically any stationary process can be represented as an infinite-order autoregression,

$$X_t = \epsilon_t + \sum_{j=1}^{\infty} \beta_j X_{t-j} \tag{23.59}$$

[26] For instance, Martindale (1990); see discussion at `http://bactra.org/weblog/666.html`.

where the "innovations" ϵ_t are serially uncorrelated, and uncorrelated with previous X's. This is matched by another result which says that basically any stationary process can be represented as an infinite-order moving average process,

$$X_t = m_t + \sum_{j=0}^{\infty} \theta_j Z_{t-j} \tag{23.60}$$

where the Z's are serially uncorrelated, and m_t is a *deterministic* linear combination of previous m's (hence the lower-case letter).

These two results, about representing stationary processes as either AR(∞) or MA(∞) processes, are called the "Wold decomposition".

Since infinite series of parameters are not very useful to the practicing statistician, people had the bright idea of trying to combine the AR and the MA parts, to get an ARMA(p, q) model:

$$\begin{aligned} X_t &= \alpha + \beta_1 X_{t-1} + \ldots + \beta_p X_{t-p} && (23.61)\\ &\quad + Z_t + \theta_1 Z_{t-1} + \ldots + \theta_q Z_{t-q} \\ &= \alpha + \beta \cdot X_{t-p:t-1} + \theta \cdot Z_{t-q:t} && (23.62)\\ & && (23.63) \end{aligned}$$

where α is an intercept, β is the vector of autoregressive parameters, θ is the vector of moving average parameters (including, by convention, $\theta_0 = 1$). Figure 23.25 illustrates the dependence structure for an ARMA(1, 1) model.

Estimation of ARMA models is complicated by the fact that the β we want here is *not* the β we'd get from just regressing X_t on $X_{t-p:t-1}$. The reason for this is evident from Figure 23.25: if we condition X_4 on, say, X_2 and X_3, there is an unblocked back-door path, $X_3 \leftarrow Z_3 \rightarrow X_4$. Indeed, by conditioning on X_3, we open a back-door path, $X_2 \leftarrow Z_2 \rightarrow X_3 \leftarrow Z_3 \rightarrow X_4$. More algebraically, Eq. 23.62 clearly implies that

$$X_t = \alpha + \beta \cdot X_{t-p:t-1} + \varepsilon_t \tag{23.64}$$

where ε_t has mean zero, but is also serially correlated, and correlated with X_t. But

$$X_t - \mathbb{E}\left[X_t | X_{t-p:t-1}\right] \tag{23.65}$$

is always uncorrelated with $X_{t-p:t-1}$, by the general properties of expectations. So whatever we'd get from a pure autoregression can't be the β of the ARMA(p, q), because plugging in that β will give correlated residuals.

There are, however, ingenious ways to get around this issue. The key trick is that X_t is (assumed to be) a *deterministic* function of $X_{t-p:t-1}$ and of $Z_{t-q:t}$. If we knew everything up to time $t-1$, our prediction for X_t would be

$$\widehat{\mu}_t = \alpha + \beta \cdot X_{t-p:t-1} + \sum_{j=1}^{q} \theta_j Z_{t-j} \tag{23.66}$$

and so

$$Z_t = X_t - \widehat{\mu}_t \tag{23.67}$$

One way to do the estimation, then, is to begin with a purely autoregressive model, and use it, via Eqs. 23.66–23.67, to get initial estimates of the Z's. If we know all the Z's, we can estimate α, β and θ by ordinary regression. Plugging those parameters back in to the equations gives updated Z's, and so forth (Durbin, 1960; Hannan and Rissanen, 1982). This is not the only way to do it, particularly if you are willing to assume the Z's are Gaussian, but the details of the various schemes are thoroughly covered in standard time series texts, e.g., Shumway and Stoffer (2000), and not worth rehashing here.

The biggest reason I do not give much space to ARMA models is that, despite their popularity, I have rarely seen them work well on real data sets. Leaving to one side Gaussian-noise assumptions, the validity of the Wold decomposition at *infinite* order does not really give any mathematical reason to expect that ARMA(p, q) models *should* work.

Finally, it's worth noting that ARMA models can be seen as a special case of the general linear state-space model, where the state S_t keeps track of all the necessary information, namely previous values of X_t and of Z_t. There are actually (at least) two ways to do this; Thiesson *et al.* (2004) describes one which is more efficient than the most obvious procedure.

23.9.3.3 Regime Switching

Hidden-state models give us another way of dealing with apparent non-stationarity, in addition to change-points and long memory processes (§23.8), namely **regime switching**. The idea is that there observed time series is in some sense driven or controlled by a discrete latent variable, the **regime**, and can show very different dynamics in different regimes. The regime itself evolves according to its own dynamics, often taken to be Markovian. If every regime has a high probability of transitioning to itself, we will see long stretches of time where the observables seem to follow one stationary process, punctuated by rare but rapid transitions to what looks like a realization of a different stationary process. If the Markov chain for regimes is stationary, the over-all process will also be stationary, but one would, so to speak, need to look over very long time scales to see it.

23.9.3.4 Noisily-Observed Dynamical Systems

23.9.4 State Estimation

If we want to know what the latent states are, we need to estimate them from the observables. We might be interested in those states for their own sake, or might need them as part of our statistical analysis. In the jargon, if we estimate S_n, the state at time n, using only observations made up to that time, $X_{1:n}$, then we are doing **filtering**. If, on the other hand, we estimate the whole sequence of states, $S_{1:n}$, from the whole sequence of observations, then we are doing **smoothing**[27]. The key difference is that in smoothing, our estimate of S_t, $t < n$, is also informed by later observations, $X_{t+1:n}$.

[27] These terms got fixed very early, when the best way to do state estimation was, in fact, to apply linear smoothers (Wiener, 1949).

Whether we are doing filtering or smoothing, there is a straightforward formal solution to the state-estimation problem, which arises from basic probability. We start with the likelihood of the observable sequence given the (hypothetical) sequence of latent states, and then use Bayes's rule:

$$p(x_{1:n}|s_{1:n}) = \prod_{t=1}^{n} p(x_t|s_t) \tag{23.68}$$

$$p(x_{1:n}) = \sum_{s_{1:n}} p(x_{1:n}|s_{1:n})p(s_{1:n}) \tag{23.69}$$

$$p(s_{1:n}|x_{1:n}) = \frac{p(x_{1:n}|s_{1:n})p(s_{1:n})}{p(x_{1:n})} \tag{23.70}$$

$$= \frac{p(x_{1:n}|s_{1:n})p(s_{1:n})}{\sum p(x_{1:n}|s_{1:n})p(s_{1:n})} \tag{23.71}$$

If the states are Markovian, we have in addition

$$p(s_{1:n}) = p(s_1)\prod_{t=2}^{n} p(s_t|s_{t-1}) \tag{23.72}$$

Putting it all together,

$$p(s_1^n|x_1^n) = \frac{\prod_{t=1}^{n} p(x_t|s_t)p(s_t|s_{t-1})}{\sum_{s'_{1:n}} \prod_{t=1}^{n} p(x_t|s_t)p(s_t|s_{t-1})} \tag{23.73}$$

If we only care about $p(s_n|x_1^n)$, a further summation takes care of that.

This is a bit easier to work with recursively. It should seem reasonable that if we know the smoothing distribution at time t, we can easily extrapolate the state one step forward in time:

$$p(s_{1:t+1}|x_{1:t}) = p(s_{1:t}|x_{1:t})p(s_{t+1}|s_{1:t}) \tag{23.74}$$

and get a predictive distribution for the next observation,

$$p(x_{t+1}|x_{1:t}) = \sum_{s_{1:t+1}} p(s_{t+1}|x_{1:t})p(x_{t+1}|s_{t+1}) \tag{23.75}$$

and then we can update the distribution over states once we see x_{t+1}:

$$p(s_{1:t+1}|x_{1:t+1}) = p(s_{1:t+1}|x_{1:t})\frac{p(x_{t+1}|s_{t+1})}{p(x_{t+1}|x_{1:t})} \tag{23.76}$$

Similarly for filtering:

$$p(s_{t+1}|x_{1:t}) = \sum_{s_t} p(s_t|x_{1:t})p(s_{t+1}|s_t) \tag{23.77}$$

and

$$p(s_{t+1}|x_{1:t+1}) = p(s_{t+1}|x_{1:t})\frac{p(x_{t+1}|s_{t+1})}{p(x_{t+1}|x_{1:t})} \tag{23.78}$$

(See Exercise 23.9.)

I said that this was just a "formal" solution to the problem, i.e., not a real solution. Eq. 23.73 is unpleasant-looking, not least the prospect of calculating the denominator — and it generally *is* hard to actually calculate. In the limited special case of linear, Gaussian state-space models (§23.9.3.1), there is a closed-form solution, given by what is called the "Kalman filter". For hidden Markov models where both X and S are discrete, we can use the EM algorithm, which in fact was first developed for just this use (Baum *et al.*, 1970), and so is sometimes called the "Baum-Welch algorithm" in this context[28].

In general, however, there just isn't any way of exactly applying Eq. 23.73, and one must consider approximations. Some of these are deterministic, such as using local linear approximations to an underlying nonlinear system, or exploiting the fact that $p(s_n|x_1^n)$ is often very sharply peaked around the most probable value (Koyama *et al.*, 2010).

23.9.4.1 Particle Filtering

23.9.4.2 Parameter Estimation

Much of the effort of the EM algorithm and of particle filtering goes into estimating the time-evolution of the latent state. If what we are willing to ignore that, and just focus on estimating the parameters, we can sometimes save greatly on time and effort by using techniques of simulation-based inference, basically adjusting the parameters until simulated trajectories of the model look like the data; see Chapter 24 for details. We could then always go back and estimate the states for *one* parameter value, or a range that reflects our uncertainty.

23.9.4.3 Prediction

23.10 Longitudinal Data

23.11 Multivariate Time Series

23.12 Further Reading

Shumway and Stoffer (2000) is a good introduction to conventional time series analysis, covering R practicalities. In particular, it includes both ARMA models, and the very important subject of frequency-domain methods, which I have deliberately omitted because it relies on Fourier analysis, otherwise not needed for this book.

On the history of how that standard machinery came to be standard, and why it seemed like a good idea, I strongly recommend Klein (1997), which is also just one of the best books I've seen on the history of statistics and statistical reasoning. I am not aware of a serious history of time-series analysis that goes past the 1930s.

[28] The origin of the name is curious. You may notice that Welch is not an author of Baum *et al.* (1970). That paper cites a submitted manuscript by Baum and Welch on "A statistical estimation procedure for probabilistic functions of finite Markov processes", which seems never to have been published. In his Shannon Lecture, Welch (2003) disclaims having done more than the "easy part" (p. 12) of coming up with the idea, and showing that it worked in some particular cases. This engaging lecture also gives an excellent overview of the algorithm.

Returning to textbooks, Lindsey (2004) surveys a broader range of situations than Shumway and Stoffer (2000) in less depth; it is readable, but opinionated, and I don't always agree with the opinions. (Try to contain your surprise.) Fan and Yao (2003) is a deservedly-standard reference on nonparametric time series models. The theoretical portions would be challenging for most readers of this book, but the methodology isn't, and it devotes about the right amount of space (i.e., little) to the usual linear-model theory. Douc *et al.* (2014) plays a similar role for parametric nonlinear statistical models; part II of that book in particular is a self-contained treatment of stochastic process theory, and part III of particle filters.

The best introduction to stochastic processes I know of, by a very wide margin, is Grimmett and Stirzaker (1992). However, like most textbooks on stochastic processes, it says next to nothing about how to use them as models of data. A notable exception is the excellent Guttorp (1995), which both introduces the theory of a range of highly-applicable stochastic processes, and covers their statistical inference with real scientific examples. Bartlett (1955), while similar in intent to Guttorp, is old enough that it now makes a better second book than a first.

The basic ergodic theorem in §23.2.2.1 follows a continuous-time argument in Frisch (1995), which seems to go back to Taylor (1922). Exercise 23.8 gives an extension to non-stationary processes. My general treatment of ergodicity is heavily shaped by Gray (1988) and Shields (1996).

As mentioned, the block bootstrap was introduced by Künsch (1989). Davison and Hinkley (1997, §8.2) has a clear treatment of the main flavors of bootstrap for time series; Lahiri (2003) is thorough but theoretical. Bühlmann (2002) is also useful.

On cross-validation for time series, classic references are Burman *et al.* (1994), Racine (2000). Carmack *et al.* (2009) is a recent proposal for a refinement. Lunde and Shalizi (2017) proposes a bootstrap method for putting confidence intervals on the prediction error, as an alternative to cross-validation.

ARMA models have spawned a huge number of modifications, extensions, and re-interpretations. Holan *et al.* (2010) is a recent survey of this "alphabet soup" of a lineage.

The notion of "state" used in state-space models ultimately derives from physics and from the mathematical theory of dynamical systems. The transition to state-space models in the current, statistical sense seems to have been made by engineers, who had to contend with imperfect measurements of the system and the possibility that it wads disturbed by noise, rather than evolving deterministically. See, for instance, McGee and Schmidt (1985) for an account of how state-space modeling was adopted by the US space program.

In parallel to the treatment of time series by statisticians, physicists and mathematicians developed their own tradition of time-series analysis (Packard *et al.*, 1980), where the basic models are not stochastic processes but deterministic, yet unstable, dynamical systems. The focus of this work is exactly recovering the latent state space from observables, with prediction often made by nearest-neighbor methods in the state space (sometimes called the "method of analogues" in this

literature). Perhaps the best guides to this are Abarbanel (1996); Kantz and Schreiber (2004). There are in fact very deep connections between this approach and the question of why probability theory works in the first place (Ruelle, 1991), but that's not a subject for *data analysis*.

A natural mathematical question is to ask which stochastic processes have state-space representations. It turns out that the answer is "basically all of them", and that there is a uniquely optimal representation for each original process. The key idea is to start by considering the conditional distribution over future events, $S_t \equiv \Pr(X_{t+1:\infty}|X_{-\infty:t})$. This is a well-defined random object, albeit one whose value is a distribution over the infinite sequence $X_{t+1:\infty}$. We can then inquire into the properties of the **prediction process** $\ldots, S_{t-1}, S_t, S_{t+1}, \ldots$. One can show that this is always a Markov process, that $X_{t+1:\infty} \perp\!\!\!\perp S_{-\infty:t-1}, X_{-\infty:t}|S_t$, and that this is, in several precise senses, the *simplest* possible process with both those properties. You can also show that there's a deterministic function q such that $S_{t+1} = q(S_t, X_{t+1})$. Thus, every stochastic process has a unique, optimal representation as a state-space process, where the states are predictive distributions for the original process, and this process is a chain with complete connections. This construction, or one mathematically equivalent to it, has been independent discovered by Crutchfield and Young (1989), Jaeger (2000), Knight (1975), Langford *et al.* (2009) and Littman *et al.* (2002) (that I know of). Of these, the treatment by Knight (1975) is the oldest, and most mathematically general. Shalizi and Crutchfield (2001) proves the information-theoretic optimality of the prediction process, and Shalizi (2003) extends it to spatio-temporal processes[29].

Throughout this chapter, I assumed that the time series records some variable (or variables) at regular time points, or perhaps continuously over some interval of time. Another very important kind of temporal data records the instants of time at which events happen — what are called **point processes**. (We might distinguish among several types of events, called **marked point processes**.) Guttorp (1995, ch. 5) is a good starting point; Daley and Vere-Jones (2003) is a standard reference on some of the deeper intricacies of the subject.

Exercises

23.1 1. (Easy) Prove that every strongly stationary process is also weakly (second-order) stationary.

2. A **Gaussian process** is one where the joint distribution $(X_{t_1}, X_{t_2}, X_{t_k})$ is always (multivariate) Gaussian, for any collection of indices $t_1, \ldots t_k$. Show that a weakly stationary Gaussian process is also strongly stationary.

3. (Harder) Give an example of a process which is weakly but not strongly stationary. *Hint:* By the previous sub-exercise, the example can't be Gaussian.

23.2 Write a function which takes in a time series X and makes a plot of X_{t+1} versus X_t, as in Figure 23.3. *Hint:* Use Code Example 32.

23.3 1. Prove that maximizing the log-likelihood in Eq. 23.25, under the constraints of Eq.

[29] If this paragraph is not more than you ever wanted to know, see `http://bactra.org/notebooks/prediction-process`.

23.26, leads to the MLE in Eq. 23.27. *Hint:* Use Lagrange multipliers, and solve for the value of the multipliers.

2. Transition rates, being probabilities, must be non-negative, $p_{ij} \geq 0$. Explain why you do not need to add yet more Lagrange multipliers to enforce this constraint.
3. Eq. 23.27 presumes that the elements p_{ij} in the $\mathbf{p}$ matrix can vary independently (subject to the constraints). Suppose instead that they are all functions of a lower-dimensional parameter vector θ. Find an expression for the MLE of θ. Do you still need the Lagrange multipliers?

23.4 In Eq. 23.40, assume that $m(x)$ has to be a linear function, $m(x) = \beta \cdot x$. Solve for the optimal β in terms of $\mathbf{y}$, $\mathbf{x}$, and $\mathbf{\Gamma}$. This "generalized least squares" (GLS) solution should reduce to ordinary least squares when $\Gamma = \sigma^2\mathbf{I}$.

23.5 If $Z_t = Z_{t-1} + \epsilon_t$, with ϵ_t IID, prove that Z_t is not stationary. *Hint:* consider $\mathbb{V}[Z_t]$.

23.6 Start with `rblockboot` from Code Example 34.

1. Modify the function to perform the circular block bootstrap. (*Hint:* Extend `ts`.)
2. Modify the function to work with multivariate time series, given as an array with time points as the rows and variables as the columns. Ensure that the *same* blocks are used for all variables, to preserve dependencies across them.
3. Modify the function to work with multivariate time series, given as a collection of univariate time series. Again, make sure the same blocks are used for all series. (*Hint:* Reduce to the previous sub-exercise.)

23.7 Suppose that X_i are IID, but we difference them and so look at $Y_i = X_i - X_{i-1}$. Find the autocovariance function of the Y series, in terms of the moments of the X_i.

23.8 *A non-stationary ergodic theorem* Suppose that the X_t are non-stationary, but they all have finite (not necessarily equal) means $\mathbb{E}[X_t]$, and finite covariances $\mathrm{Cov}[X_t, X_s]$. Define

$$m_n \equiv \frac{1}{n}\sum_{t=1}^{n} \mathbb{E}[X_t] \tag{23.79}$$

and

$$V_n \equiv \sum_{t=1}^{n}\sum_{s=1}^{n} \mathrm{Cov}[X_t, X_s] \tag{23.80}$$

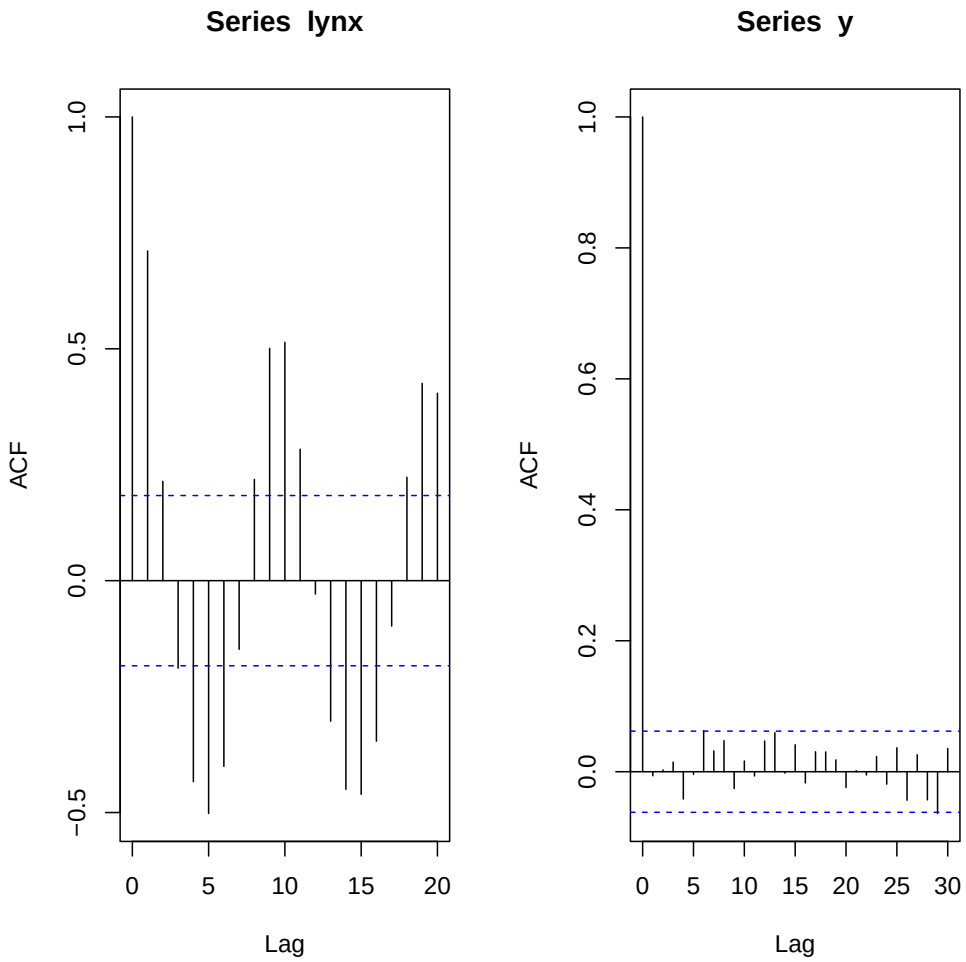
Show that if $V_n = o(n^2)$, then

$$\mathbb{E}\left[\left(m_n - \overline{X}_n\right)^2\right] \to 0 \tag{23.81}$$

and so that $\overline{X}_n \to m_n$. Does this result imply Eq. 23.15 under the conditions of §23.2.2.1? Could you deduce this result from Eq. 23.15?

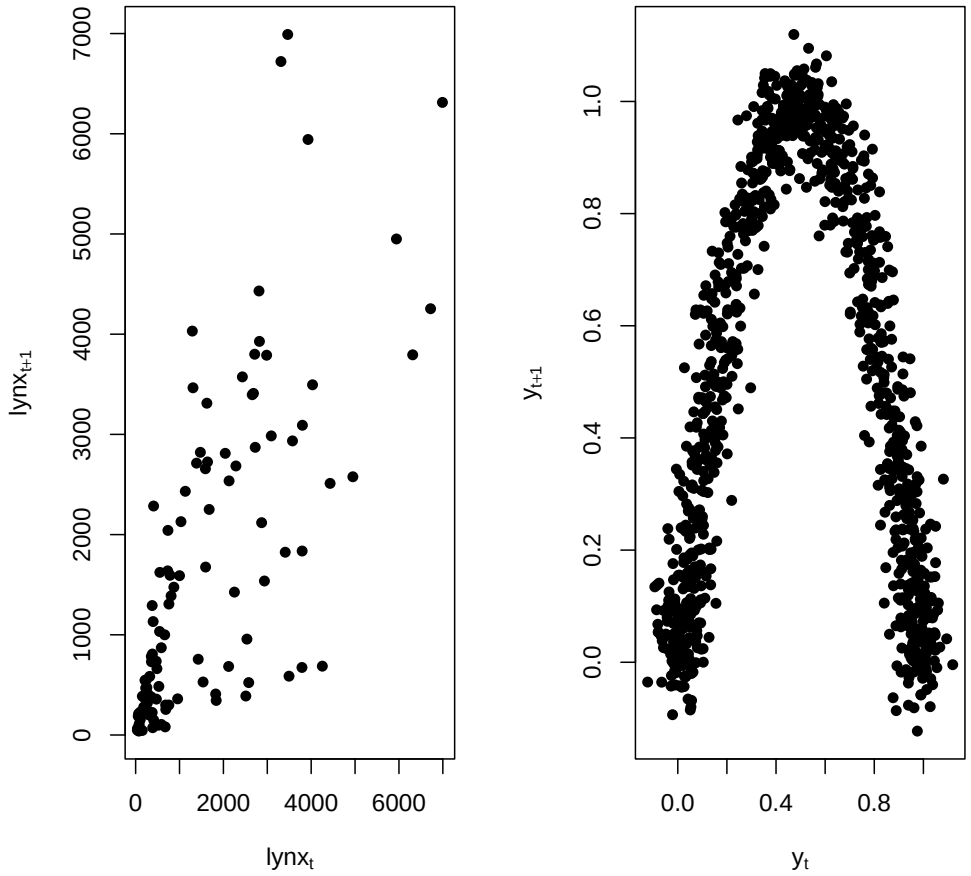
23.9 *Recursive equations for state estimation*

1. Derive Eq. 23.74. *Hint:* First show that $S_{t+1} \perp\!\!\!\perp X_{1:t} | S_t$.
2. Derive Eq. 23.75. *Hint:* First show that $X_{t+1} \perp\!\!\!\perp X_{1:t} | S_t$.
3. Derive Eq. 23.76.
4. Derive Eq. 23.77.
5. Derive Eq. 23.78.




```
par(mfrow = c(1, 2))
acf(lynx)
acf(y)
par(mfrow = c(1, 1))
```

Figure 23.2 Autocorrelation functions of the lynx data (above) and the simulation (below). The `acf` function plots the autocorrelation function as an automatic side-effect; it actually returns the actual value of the autocorrelations, which you can capture. The 95% confidence interval around zero is computed under Gaussian assumptions which shouldn't be taken too seriously, unless the sample size is quite large, but are useful as guides to the eye.




```
par(mfrow = c(1, 2))
plot(lag0 ~ lag1, data = design.matrix.from.ts(lynx, 1), xlab = expression(lynx[t]),
    ylab = expression(lynx[t + 1]), pch = 16)
plot(lag0 ~ lag1, data = design.matrix.from.ts(y, 1), xlab = expression(y[t]), ylab = expression(y[t
    1]), pch = 16)
par(mfrow = c(1, 1))
```

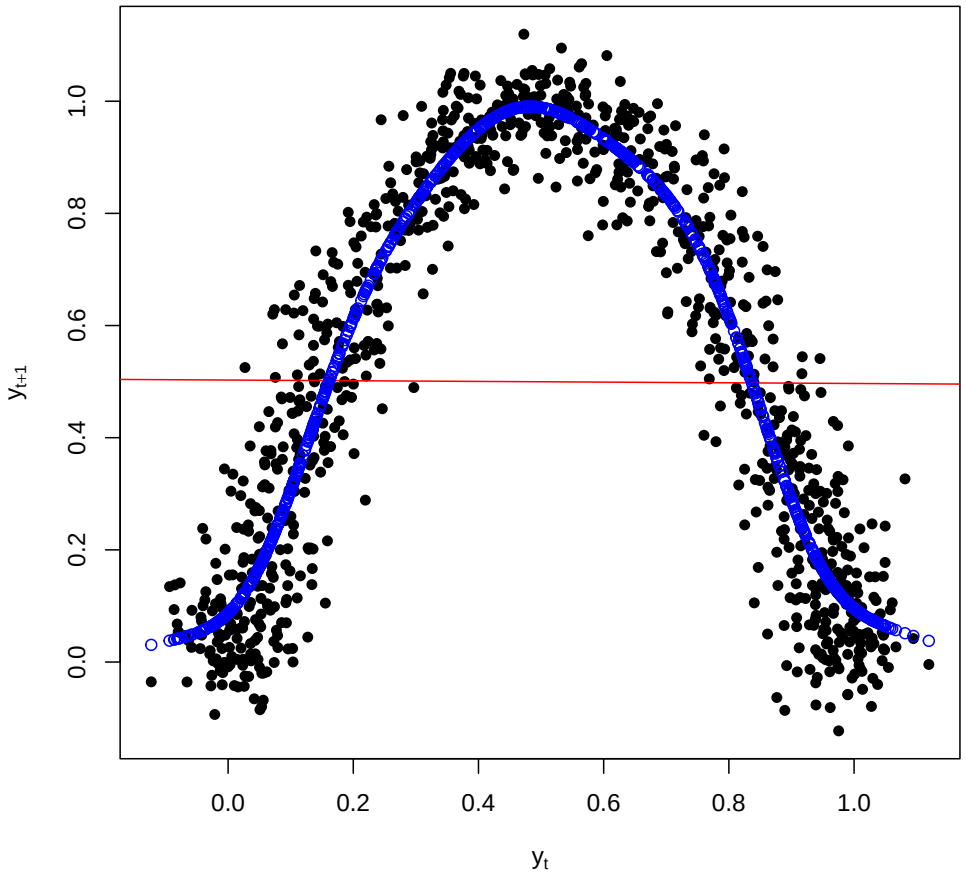
Figure 23.3 Plots of X_{t+1} versus X_t, for the lynx (left) and the simulation (right); see Exercise 23.2. Note that even though the correlation between successive iterates is next to zero for the simulation, there is clearly a lot of dependence (see Appendix **??**).

t	x
1821	269
1822	321
1823	585
1824	871
1825	1475
1826	2821
1827	3928
1828	5943
1829	4950
...	

$\Rightarrow$

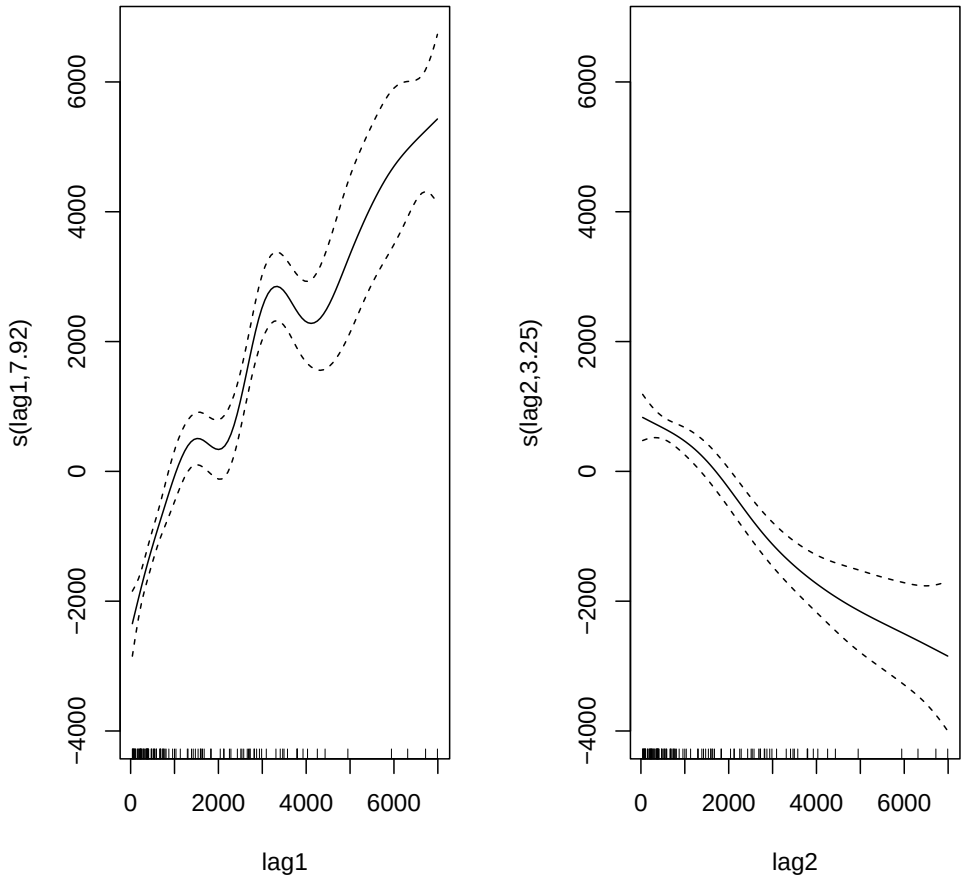
lag0	lag1	lag2	lag3
871	585	321	269
1475	871	585	321
2821	1475	871	585
3928	2821	1475	871
5943	3928	2821	1475
4950	5943	3928	2821
...			

Figure 23.4 Turning a time series (here, the beginning of `lynx`) into a regression-suitable matrix.



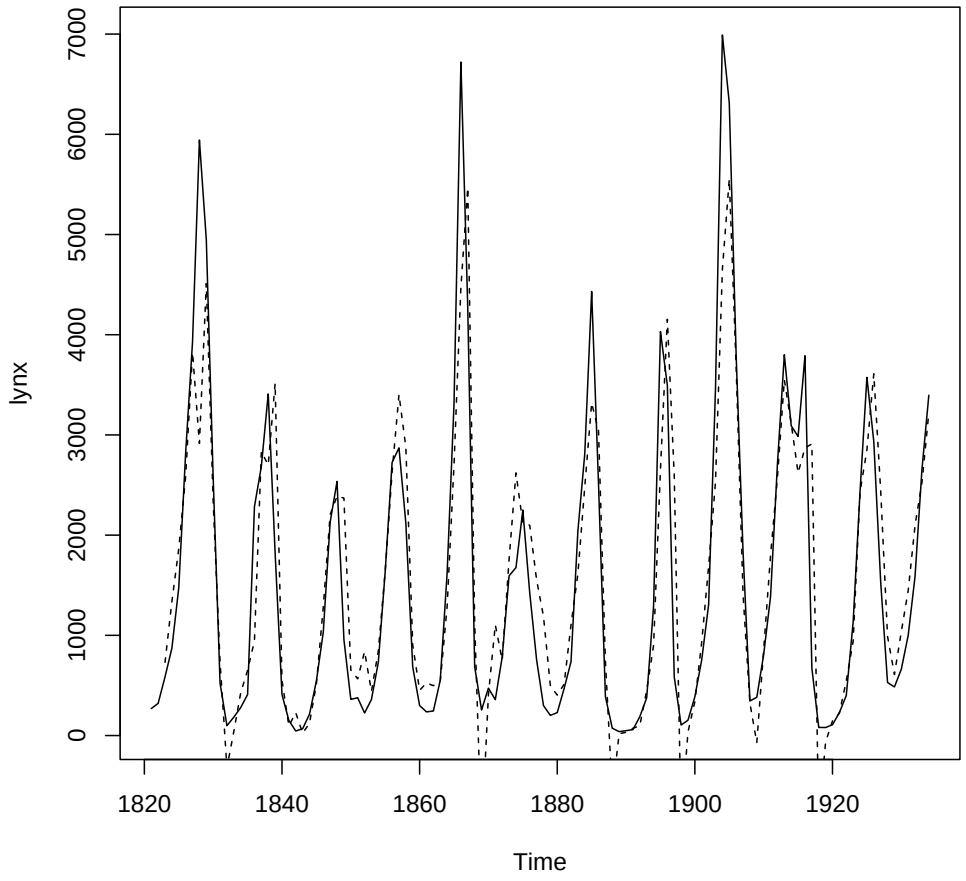

```
plot(lag0 ~ lag1, data = design.matrix.from.ts(y, 1), xlab = expression(y[t]), ylab = expression(y[t
    1]), pch = 16)
abline(lm(lag0 ~ lag1, data = design.matrix.from.ts(y, 1)), col = "red")
yaar1 <- aar(y, order = 1)
points(y[-length(y)], fitted(yaar1), col = "blue")
```

Figure 23.5 Plotting successive values of the artificial time series against each other, along with the linear regression, and a spline curve (see below for the `aar` function, which fits additive autoregressive models; with `order=1`, it just fits a spline.



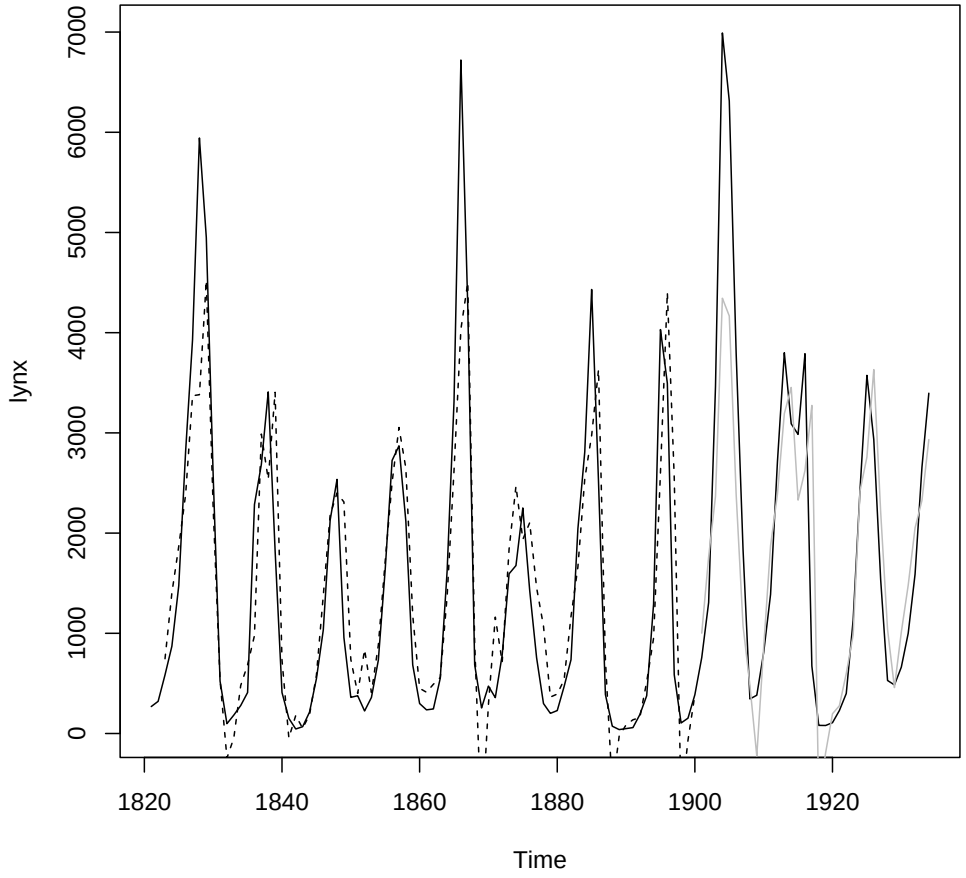

```
plot(lynx.aar2, pages = 1)
```

Figure 23.6 Partial response functions for the second-order additive autoregression model of the lynx. Notice that a high count last year predicts a higher count this year, but a high count *two* years ago predicts a lower count this year. This is the sort of alternation which will tend to drive oscillations.



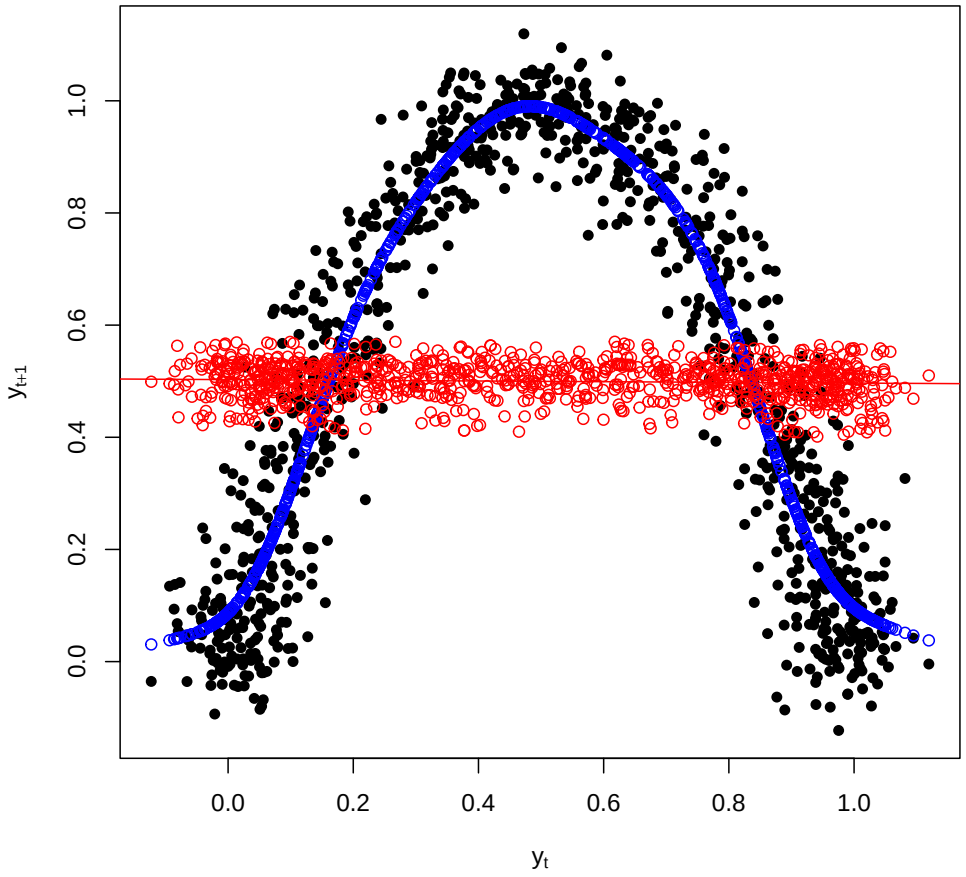

```
plot(lynx)
lines(1823:1934, fitted(lynx.aar2), lty = "dashed")
```

Figure 23.7 Actual time series (solid line) and predicted values (dashed) for the second-order additive autoregression model of the lynx. The match is quite good, but of course every one of these points was used to learn the model, so it's not quite as impressive as all that. (Also, the occasional prediction of a negative number of lynxes is less than ideal.)



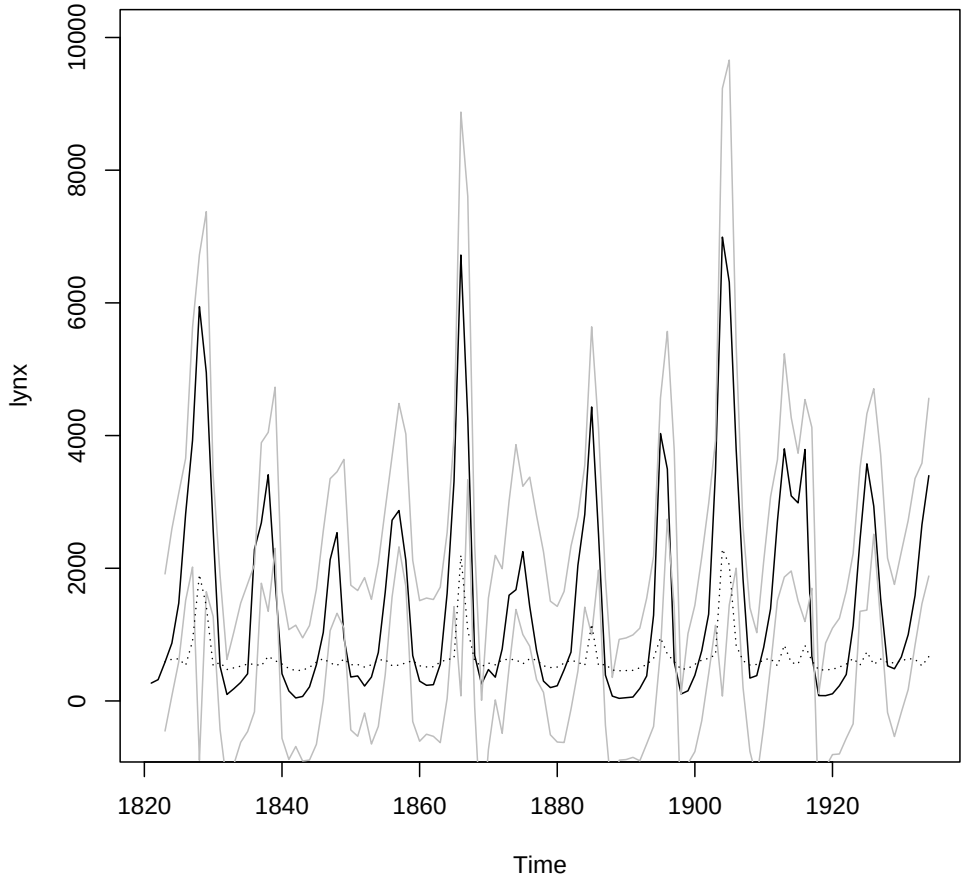

```
lynx.aar2b <- aar(lynx[1:80], 2)
out.of.sample <- design.matrix.from.ts(lynx[-(1:78)], 2)
lynx.preds <- predict(lynx.aar2b, newdata = out.of.sample)
plot(lynx)
lines(1823:1900, fitted(lynx.aar2b), lty = "dashed")
lines(1901:1934, lynx.preds, col = "grey")
```

Figure 23.8 Out-of-sample forecasting. The same model specification as before is estimated on the first 80 years of the lynx data, then used to predict the remaining 34 years. Solid black line, data; dashed line, the in-sample prediction on the training data; grey lines, predictions on the testing data. The RMS errors are 723 lynxes/year in-sample, 922 lynxes/year out-of-sample.




```
library(tseries)
yar8 <- arma(y, order = c(8, 0))
points(y[-length(y)], fitted(yar8)[-1], col = "red")
```

Figure 23.9 Adding the predictions of an eighth-order linear AR model (red dots) to Figure 23.5. We will see the `arma` function in more detail in §23.9.3.2; for now, it's enough to know that when the second component of its `order` argument is 0, it estimates and fits a linear AR model.




```
plot(lynx, ylim = c(-500, 10000))
sd1 <- sqrt(fitted(lynx.condvar1))
lines(1823:1934, fitted(lynx.aar2) + 2 * sd1, col = "grey")
lines(1823:1934, fitted(lynx.aar2) - 2 * sd1, col = "grey")
lines(1823:1934, sd1, lty = "dotted")
```

Figure 23.10 The lynx data (black line), together with the predictions of the additive autoregression ± 2 conditional standard deviations. The dotted line shows how the conditional standard deviation changes over time; notice how it ticks upwards around the big spikes in population.

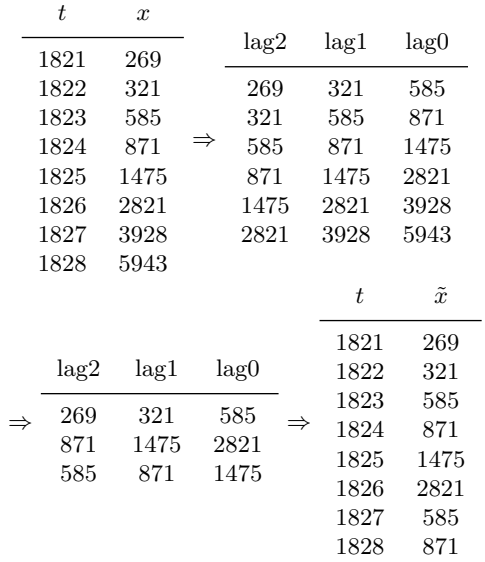
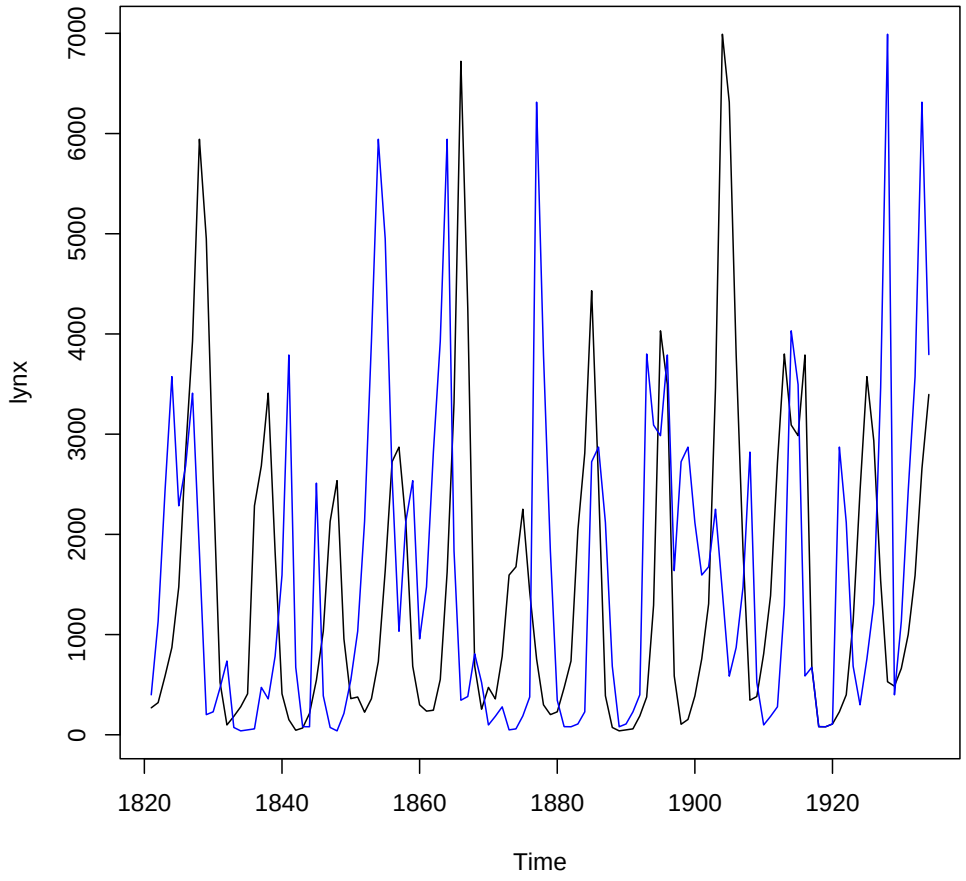
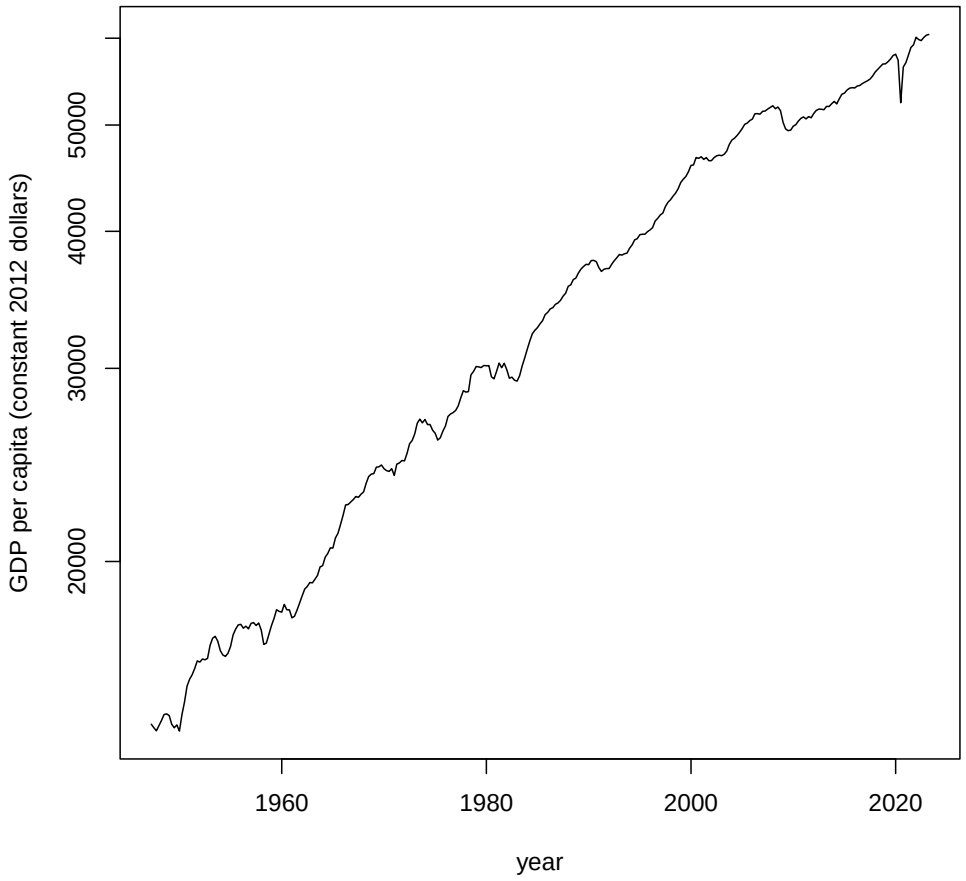


Figure 23.11 Scheme for block bootstrapping: turn the time series (here, the first eight years of `lynx`) into blocks of consecutive values; randomly resample enough of these blocks to get a series as long as the original; then string the blocks together in order. See Code Example 34 for code.



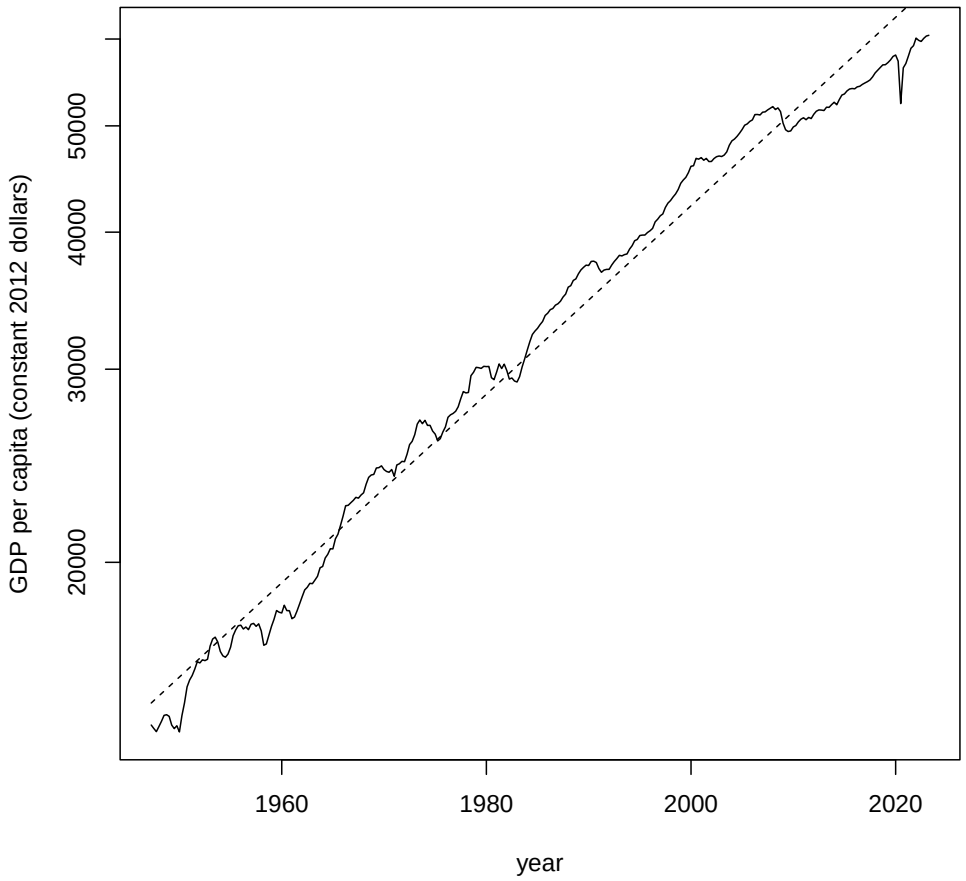

```
plot(lynx)
lines(1821:1934, rblockboot(lynx, 4), col = "blue")
```

Figure 23.12 The lynx time series, and one run of resampling it with a block bootstrap, block length = 4.



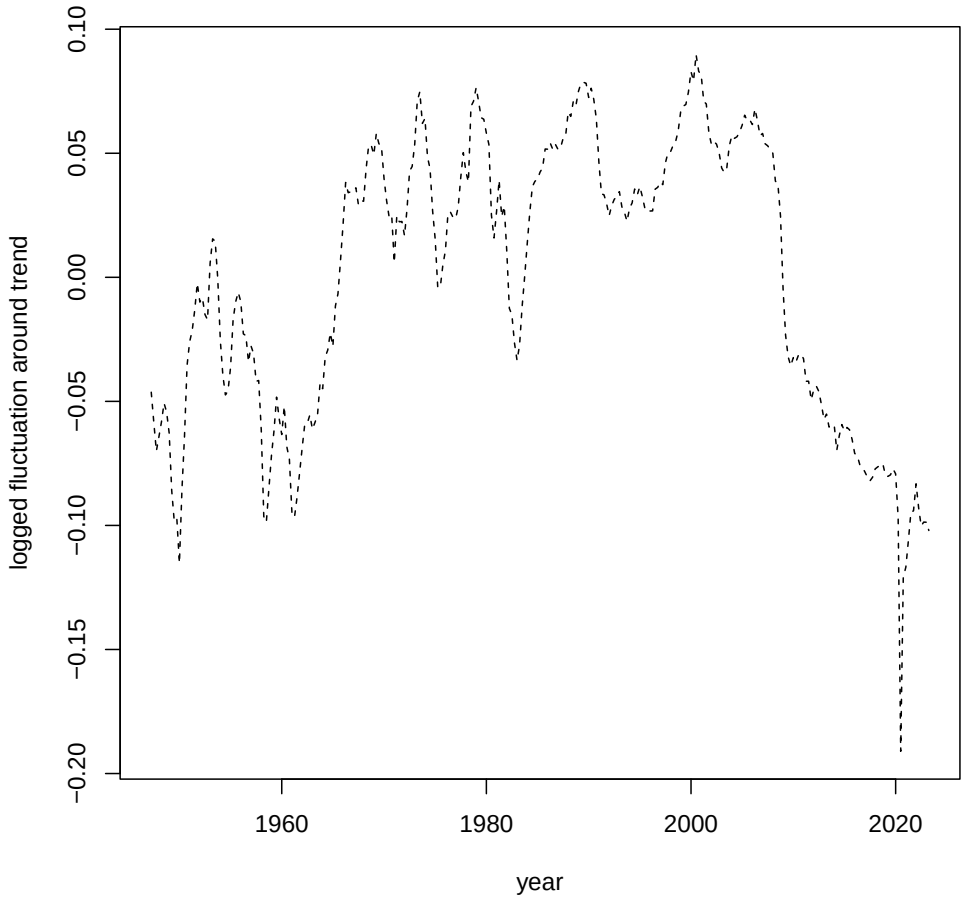

```
library(pdfetch)
gdppc.fred <- pdfetch_FRED("A939RX0Q048SBEA")
library(xts)
library(lubridate)
gdppc <- data.frame(year = decimal_date(index(gdppc.fred)), y = as.numeric(gdppc.fred))
plot(gdppc, log = "y", type = "l", ylab = "GDP per capita (constant 2012 dollars)")
```

Figure 23.13 US GDP per capita, adjusted for inflation (consumer price index deflator), with a log scale on the vertical axis. (The values were initially recorded in the file in millions of dollars per person per year, hence the correction.)



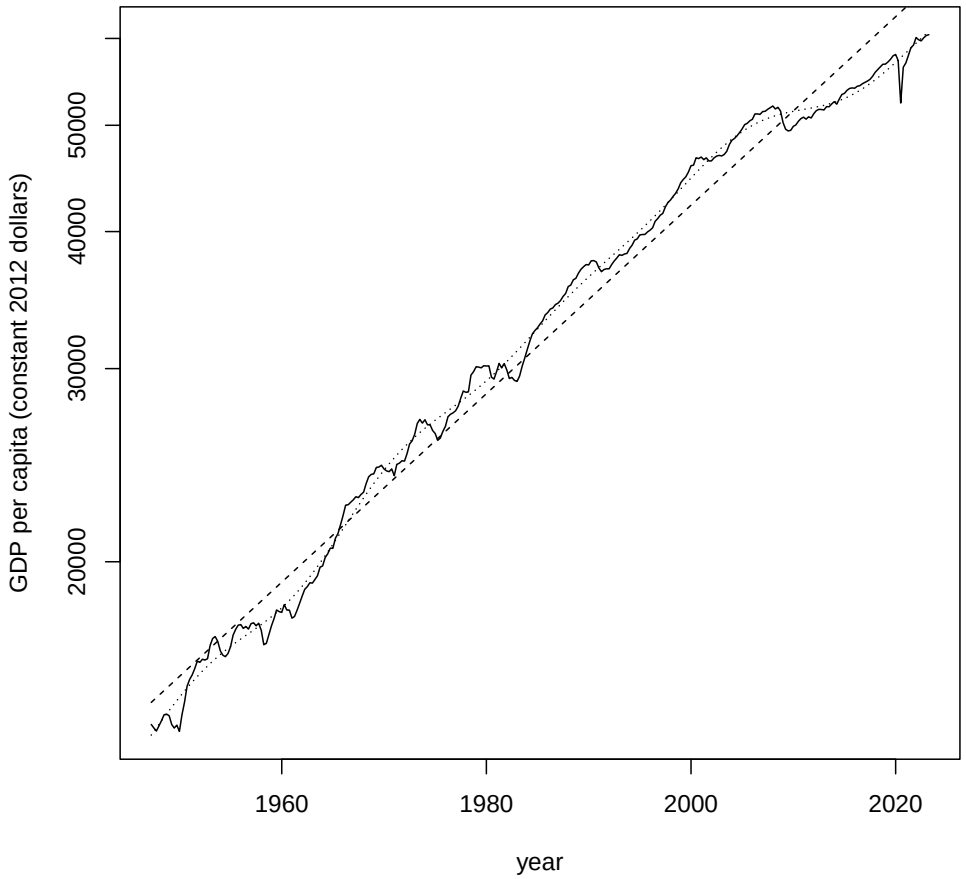

```
gdppc.exp <- lm(log(y) ~ year, data = gdppc)
beta0 <- exp(coefficients(gdppc.exp)[1])
beta <- coefficients(gdppc.exp)[2]
curve(beta0 * exp(beta * x), lty = "dashed", add = TRUE)
```

Figure 23.14 As in Figure 23.13, but with an exponential trend fitted.



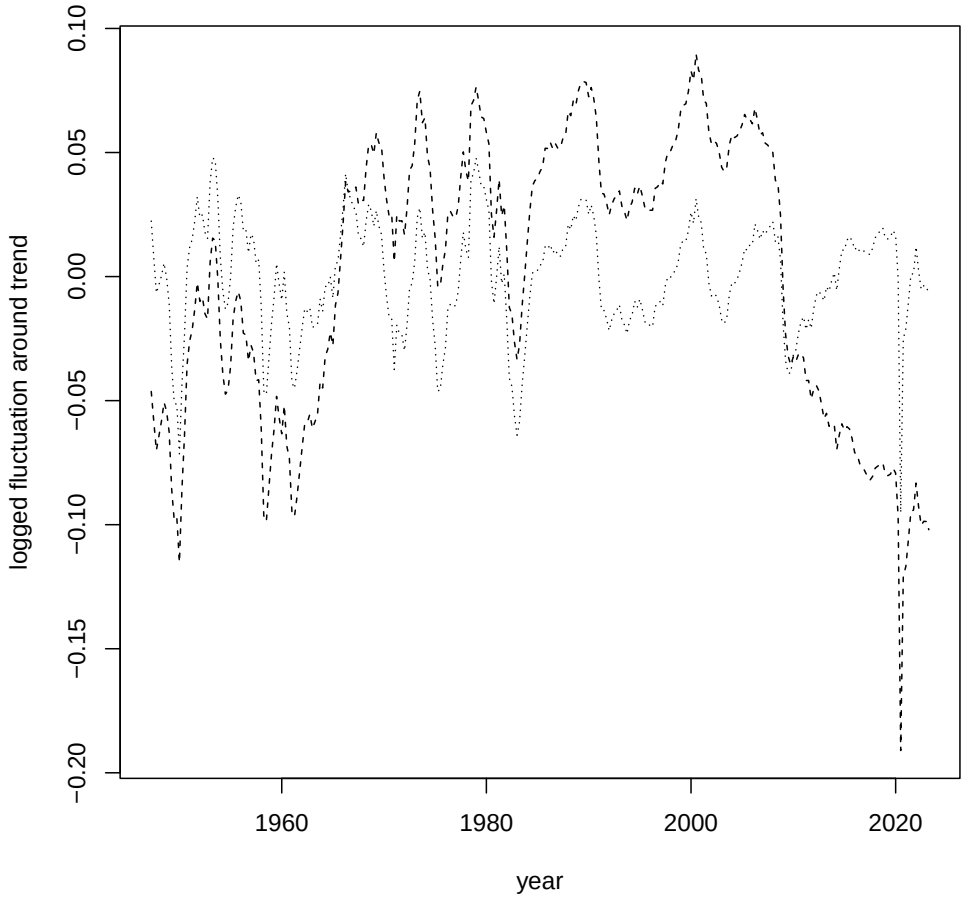

```
plot(gdppc$year, residuals(gdppc.exp), xlab = "year", ylab = "logged fluctuation around trend",
    type = "l", lty = "dashed")
```

Figure 23.15 The hopefully-stationary fluctuations around the exponential growth trend in Figure 23.14. Note that these are log $\frac{Y_t}{\hat{\beta}_0 e^{\hat{\beta} t}}$, and so unitless.



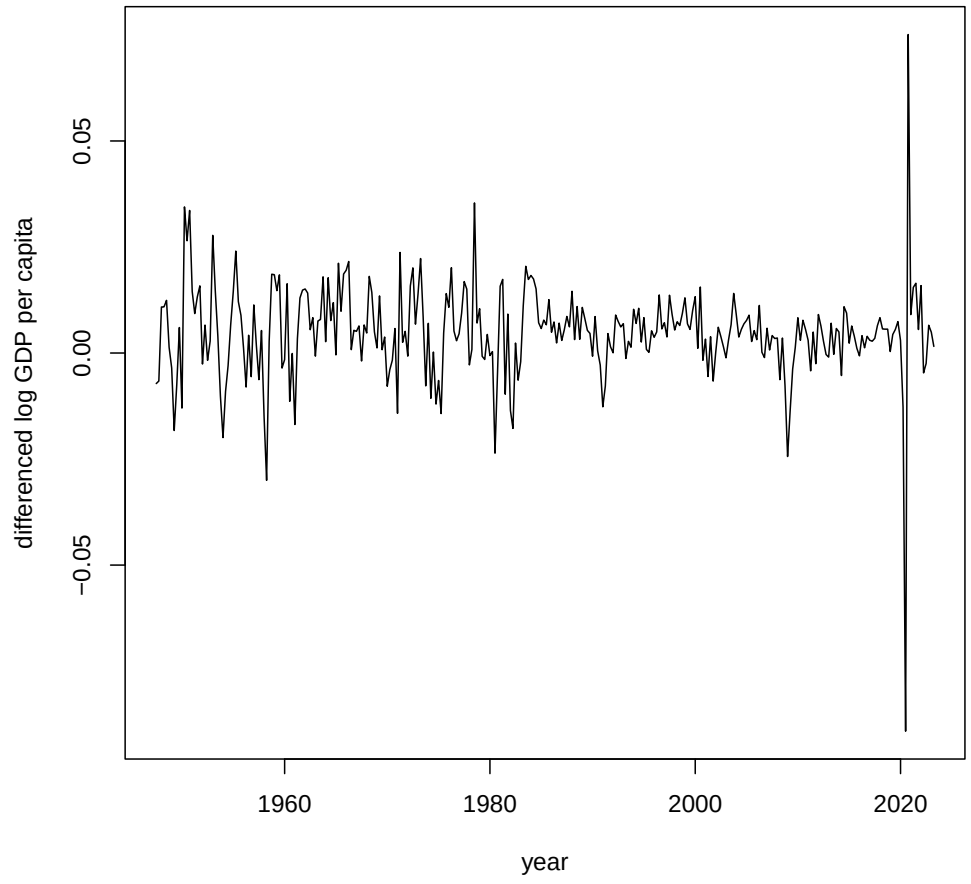

```
gdp.spline <- fitted(gam(y ~ s(year), data = gdppc))
lines(gdppc$year, gdp.spline, lty = "dotted")
```

Figure 23.16 Figure 23.14, but with the addition of a spline curve for the time trend (dotted line). This is, perhaps unsurprisingly, not all *that* different from the simple exponential-growth trend.



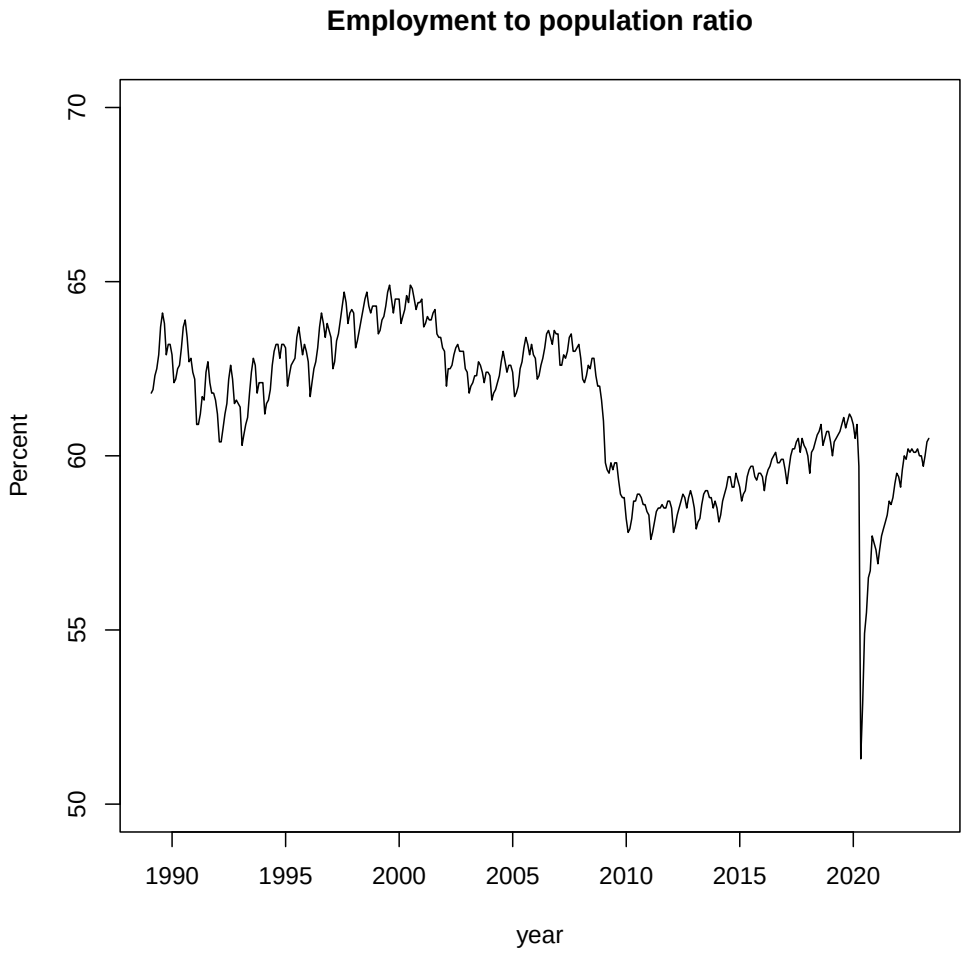

```
lines(gdppc$year, log(gdppc$y/gdp.spline), xlab = "year", ylab = "logged fluctuations around trend",
    lty = "dotted")
```

Figure 23.17 Adding the logged deviations from the spline trend (dotted) to Figure 23.15.



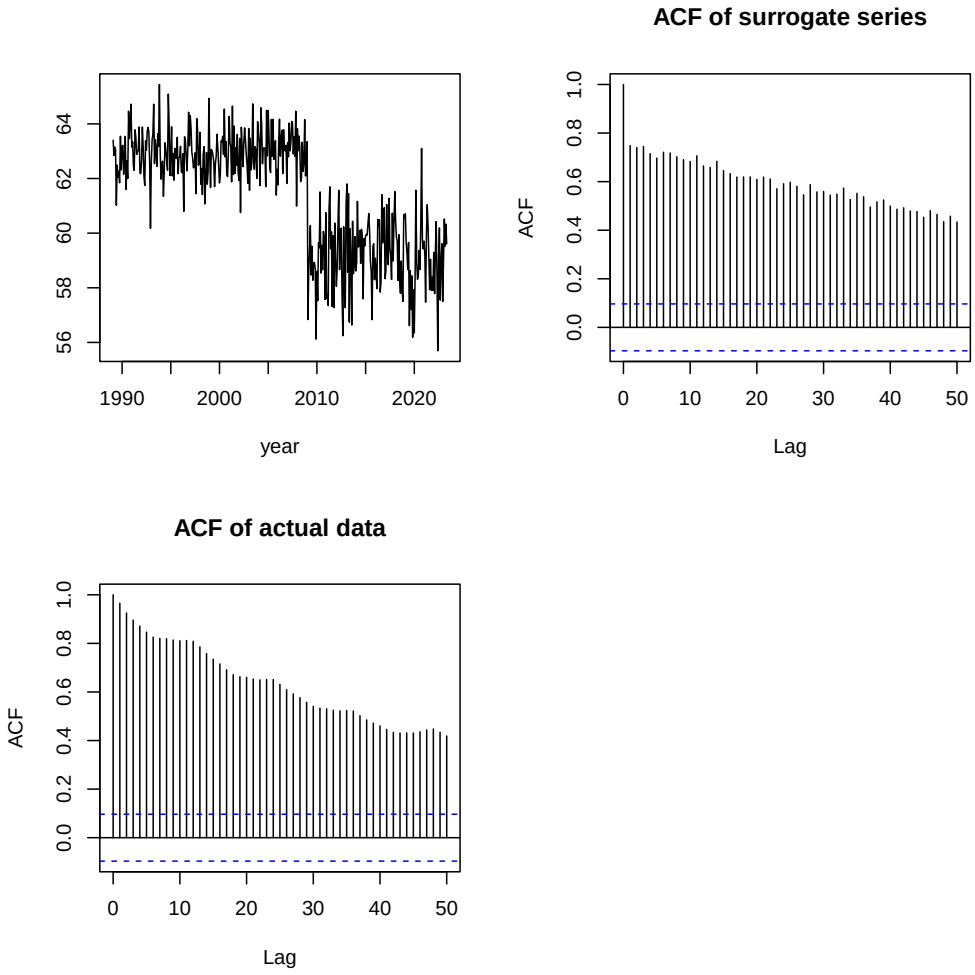

```
plot(gdppc$year[-1], diff(log(gdppc$y)), type = "l", xlab = "year", ylab = "differenced log GDP per
```

Figure 23.18 First differences of log GDP per capita, i.e., the year-to-year growth rate of GDP per capita.




```
epr.fred <- pdfetch_FRED("LNU02300000")
epr <- data.frame(year = decimal_date(index(epr.fred)), epr = as.numeric(epr.fred))
epr <- epr[epr$year > 1989, ]
plot(epr, ylab = "Percent", ylim = c(50, 70), main = "Employment to population ratio",
    type = "l")
```

Figure 23.19 Monthly employment to population ratio for the US, in percent, without seasonal adjustment, from 1990 forward. (Source: series LNU02300000 from FRED, `https://fred.stlouisfed.org/series/LNU02300000`.)




```
par(mfrow = c(2, 2))
before_crash <- (epr$year < 2009)
after_crash <- (epr$year >= 2009)
epr_before <- epr$epr[before_crash]
epr_after <- epr$epr[after_crash]
pre <- rnorm(sum(before_crash), mean(epr_before), sd(epr_before))
post <- rnorm(sum(after_crash), mean(epr_after), sd(epr_after))
change <- data.frame(year = epr$year, epr = c(pre, post))
plot(change, ylab = "", type = "l")
acf(change$epr, lag.max = 50, main = "ACF of surrogate series")
acf(epr$epr, lag.max = 50, main = "ACF of actual data")
par(mfrow = c(1, 1))
```

Figure 23.20 A time series with a change-point. Before and after the change point, the series is an IID sequence of Gaussians, but both the expected value and the variance switch at the change-point. (These are matched to the employment-population ratio's values up to 2008 and after 2008.) The middle panel shows the resulting autocorrelation function. The bottom panel shows the actual ACF of the employment-population ratio. There is more correlation in the data than the change-point alone can account for, but it comes close.

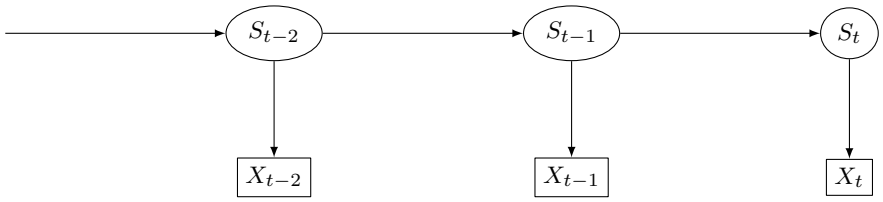


Figure 23.21 DAG for hidden Markov models. The current state S_t is the only parent of the current observation X_t *and* the only parent of the next state S_{t+1}. Thus the S's are Markovian, the X's are not, $X_t \perp\!\!\!\perp S_{-\infty:t-1}, X_{-\infty:t-1}|S_t$, *and* $X_{t+1:\infty}, S_{t+1:\infty} \perp\!\!\!\perp X_t|S_t$.

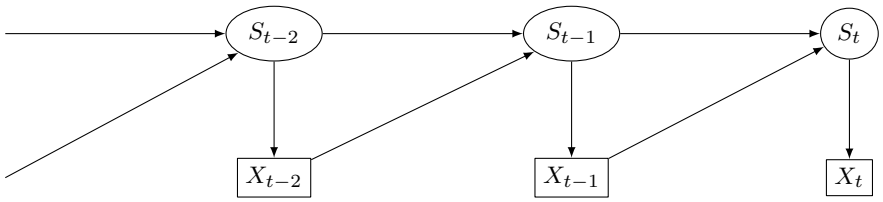


Figure 23.22 DAG for a chain with complete connections. The current state S_t is the only parent of the current observation X_t, and those two together are the only parents of the next state S_{t+1}. (In fact, they're usually assumed to fix S_{t+1} deterministically.) The S's are Markovian, the X's are not, $X_t \perp\!\!\!\perp S_{-\infty:t-1}, X_{-\infty:t-1}|S_t$, *and* $X_{t+1:\infty}, S_{t+1:\infty} \not\perp\!\!\!\perp X_t|S_t$.

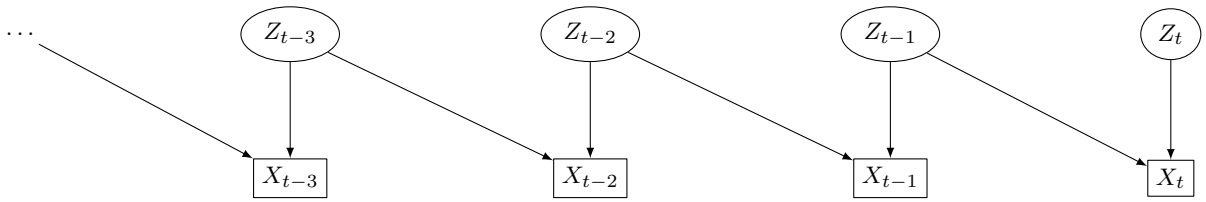
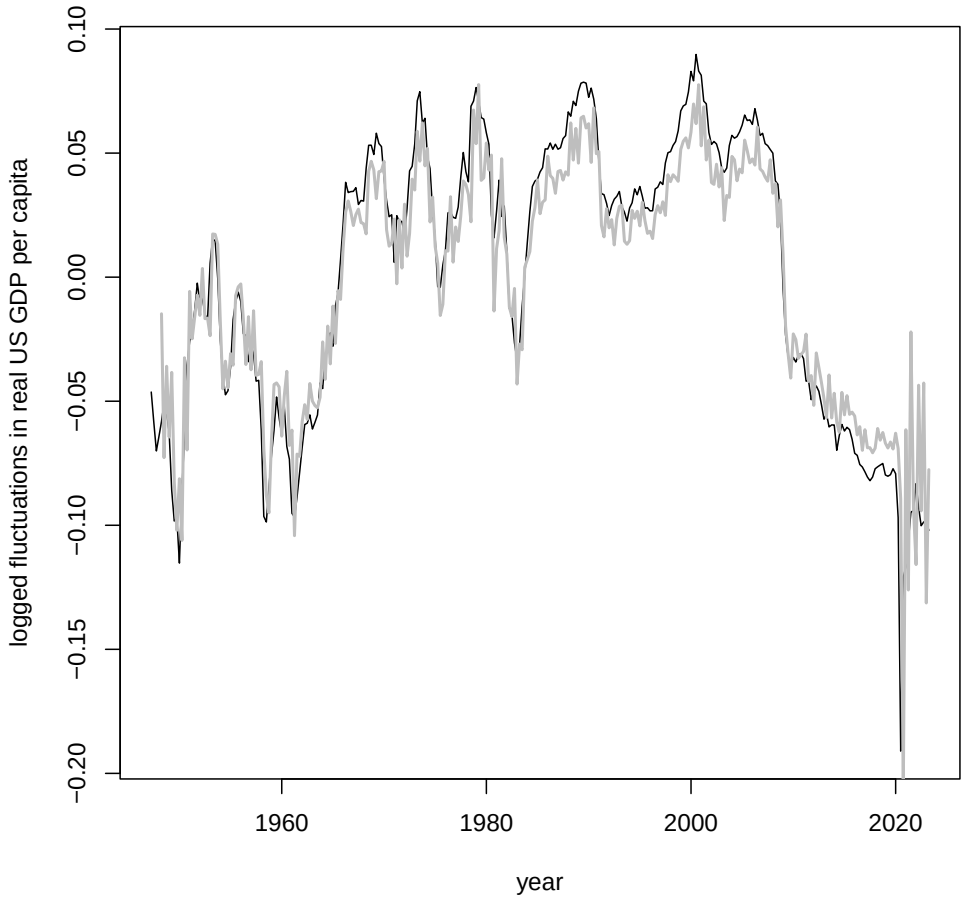


Figure 23.23 The DAG for a first-order moving average model.




```
gdppc.ma4 <- arma(x = residuals(gdppc.exp), order = c(0, 4))
plot(gdppc$year, residuals(gdppc.exp), type = "l", xlab = "year", ylab = "logged fluctuations in rea
lines(gdppc$year, fitted(gdppc.ma4), col = "grey", lwd = 2)
```

Figure 23.24 Logged fluctuations for the United States's GDP per capita (with exponential trend removed, as in Figure 23.15), versus a fourth-order moving average model. (Since each unit of time is a quarter, four quarters is a year.) The root-mean-squared error, in sample, is 0.018, corresponding to an R^2 of 0.71. (But you know better than to rely on R^2.

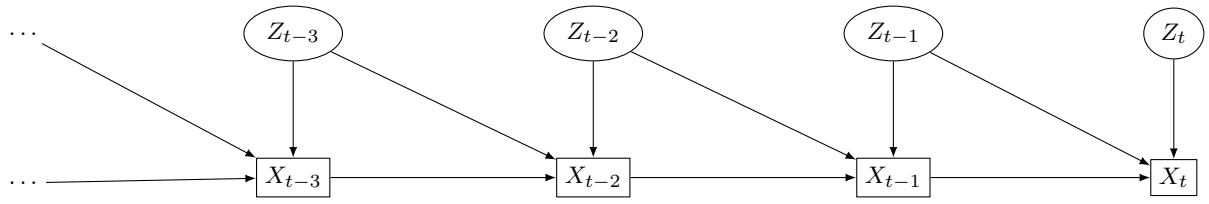


Figure 23.25 The DAG for an ARMA$(1, 1)$ model.

24

Simulation-Based Inference

Checking whether the model's simulation output looks like the data (§5.4.2) naturally suggests the idea of *adjusting* the model until it does. This becomes a way of estimating the model — in the jargon, **simulation-based inference**. All forms of simulation-based inference involve tweaking parameters of the model until the simulations *do* look like the data, but differ in what, concretely, "looking like the data" means.

24.1 The Method of Simulated Moments

The most straightforward form of simulation-based inference is the **method of simulated moments**, which builds of the **method of moments** you'll have seen in earlier statistics classes.

24.1.1 The Method of Moments

We have a model with a parameter vector θ, and pick a vector m of moments to calculate. The moments, like the expectation of any variables, are functions of the parameters,

$$m = g(\theta) \tag{24.1}$$

for some function[1] g. If that g is invertible, then we can recover the parameters from the moments,

$$\theta = g^{-1}(m) \tag{24.2}$$

The method of moments *estimator* takes the observed, sample moments $\hat{m}$, and plugs them into Eq. 24.2:

$$\widehat{\theta_{MM}} = g^{-1}(\hat{m}) \tag{24.3}$$

[1] In some situations, it's more convenient to think of moments or other expectation values as functions of the entire *distribution*, represented by the probability density function $p(x;\theta)$, not just the parameter vector θ. (This is especially true, naturally, when we're trying to do nonparametrics and don't have parameters.) A slightly old-fashioned mathematical name for a function of a function is a **functional**, so one finds phrases like "moments are functionals of the distribution", or "moments and other statistical functionals".

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

If g^{-1} is hard to calculate — if it's hard to explicitly solve for parameters from moments — we can minimize instead:

$$\widehat{\theta_{MM}} = \operatorname*{argmin}_{\theta} \|g(\theta) - \hat{m}\|^2 \tag{24.4}$$

For the minimization version, we just have to calculate moments from parameters $g(\theta)$, not vice versa. To see that Eqs. 24.3 and 24.4 do the same thing, notice that (i) the squared[2] distance $\|g(\theta) - \hat{m}\|^2 \geq 0$, (ii) the distance is only zero when the moments are matched exactly, and (iii) there is only one θ which will match the moments.

In either version, inversion or minimization, the method of moments works *statistically* because the sample moments $\hat{m}$ converge on their expectations $g(\theta)$ as we get more and more data (App. D.5). This is, generally, a consequence of the law of large numbers or ergodic theorem. indexthe usual asymptotics—seeasymptotics, the usual

It's worth noting that nothing in this argument says that m has to be a vector of *moments* in the strict sense. They could be expectations of any functions of the random variables, so long as $g(\theta)$ is invertible, we can calculate the sample expectations of these functions from the data, and the sample expectations converge. When m isn't just a vector of moments, then, we have the **generalized method of moments**.

It is also worth noting that there's a somewhat more general version of the same method, where we minimize

$$(g(\theta) - \hat{m}) \cdot \mathbf{w} \, (g(\theta) - \hat{m}) \tag{24.5}$$

with some positive-definite weight matrix $\mathbf{w}$. This can help if some of the moments are much more sensitive to the parameters than others.

24.1.2 Adding in the Simulation

All of this supposes that we know how to calculate $g(\theta)$ — that we can find the moments exactly. Even if this is too hard, however, we could always *simulate* to approximate these expectations (as in the Monte Carlo method), and try to match the simulated moments to the real ones. Rather than Eq. 24.4, the estimator would be

$$\widehat{\theta_{SMM}} = \operatorname*{argmin}_{\theta} \|\tilde{g}_{s,T}(\theta) - \hat{m}\|^2 \tag{24.6}$$

with s being the number of simulation paths and T being their size. Now consistency requires that $\tilde{g} \to g$, either as T grows or s or both, but this is generally assured by the law of large numbers, as before. Simulated method of moments estimates like this are generally more uncertain than ones which don't rely on

[2] Why squared? Basically because it makes the function we're minimizing smoother, and the optimization nicer.

simulation, since there's an extra layer of approximation, but this can be reduced by increasing s.[3]

24.1.3 An Example: Moving Average Models and the Stock Market

To give a concrete example, we will try fitting a time series model to the stock market: it's a familiar subject which interests *most* students, and we can check the method of simulated moments here against other estimation techniques.

Our data will consist of about ten year's worth of daily values for the S& P 500 stock index, previously seen in Chapter 7:

```
sp <- pdfetch_YAHOO("SPY", fields = "adjclose", from = as.Date("1993-02-09"), to = as.Date("2018-02-
sp <- diff(log(sp))
sp <- sp[-1]
```

Professionals in finance do not care so much about the sequence of **prices** P_t, as the sequence of **returns**, $\frac{P_t - P_{t-1}}{P_{t-1}}$. This is because making \$1000 is a lot better when you invested \$1000 than when you invested \$1,000,000, but 10% is 10%. In fact, it's often easier to deal with the **log returns**, $X_t = \log \frac{P_t}{P_{t-1}}$, as we do here.

The model we will fit is a **first-order moving average**, or MA(1) model (§23.9.1):

$$X_t = Z_t + \theta Z_{t-1} \tag{24.7}$$

$$Z_t \sim \mathcal{N}(0, \sigma^2) \text{ i.i.d.} \tag{24.8}$$

The X_t sequence of variables are the returns we see; the Z_t variables are invisible to us. The interpretation of the model is as follows. Prices in the stock market change in response to news that affects the prospects of the companies listed, as well as news about changes in over-all economic conditions. Z_t represents this flow of news, good and bad. It makes sense that Z_t is uncorrelated, because the relevant part of the news is only what everyone hadn't already worked out from older information[4]. However, it does take some time for the news to be assimilated, and this is why Z_{t-1} contributes to X_t. A negative contribution, $\theta < 0$, would seem to indicate a "correction" to the reaction to the previous day's news.

Mathematically, notice that since Z_t and θZ_{t-1} are independent Gaussians, X_t is a Gaussian with mean 0 and variance $\sigma^2 + \theta^2\sigma^2$. The marginal distribution of X_t is therefore the same for all t. For technical reasons[5], we can really only get sensible behavior from the model when $-1 \leq \theta \leq 1$.

[3] A common trick is to fix T at the actual sample size n, and then to increase s as much as computationally feasible. By looking at the variance of $\tilde{g}$ across different runs of the model with the same θ, one gets an idea of how much uncertainty there is in $\hat{m}$ itself, and so of how precisely one should expect to be able to match it. If the optimizer has gotten $|\tilde{g}(\theta) - \hat{m}|$ down to 0.02, and the standard deviation of $\tilde{g}$ at constant θ is 0.1, further effort at optimization is probably wasted.

[4] Nobody will ever say "What? It's snowing in Pittsburgh in *February*? Call my broker!"

[5] Think about trying to recover Z_t, if we knew θ. One might try $X_t - \theta X_{t-1}$, which is almost right, it's $Z_t + \theta Z_{t-1} - \theta Z_{t-1} - \theta^2 Z_{t-2} = Z_t - \theta^2 Z_{t-2}$. Similarly, $X_t - \theta X_{t-1} + \theta^2 X_{t-2} = Z_t + \theta^3 Z_{t-2}$, and so forth. If $|\theta| < 1$, then this sequence of approximations will converge on Z_t; if not, then not. It

There are two parameters, θ and σ^2, so we need two moments for estimation. Let's try $\mathbb{V}[X_t]$ and $\mathrm{Cov}[X_t, X_{t-1}]$.

$$\begin{aligned}
\mathbb{V}[X_t] &= \mathbb{V}[Z_t] + \theta^2 \mathbb{V}[Z_{t-1}] && (24.9)\\
&= \sigma^2 + \theta^2\sigma^2 && (24.10)\\
&= \sigma^2(1+\theta^2) \equiv v(\theta,\sigma) && (24.11)\\
\mathrm{Cov}[X_t, X_{t-1}] &= \mathbb{E}[(Z_t + \theta Z_{t-1})(Z_{t-1} + \theta Z_{t-2})] && (24.12)\\
&= \theta \mathbb{E}\left[Z_{t-1}^2\right] && (24.13)\\
&= \theta\sigma^2 \equiv c(\theta,\sigma) && (24.14)
\end{aligned}$$

We can solve the system of equations for the parameters, starting with eliminating σ^2:

$$\begin{aligned}
\frac{c(\theta,\sigma)}{v(\theta,\sigma)} &= \frac{\sigma^2\theta}{\sigma^2(1+\theta^2)} && (24.15)\\
&= \frac{\theta}{1+\theta^2} && (24.16)\\
0 &= \theta^2\frac{c}{v} - \theta + \frac{c}{v} && (24.17)
\end{aligned}$$

This is a quadratic in θ,

$$\theta = \frac{1 \pm \sqrt{1 - 4\frac{c^2}{v^2}}}{2c/v} \tag{24.18}$$

and it's easy to confirm[6] that this has only one solution in the meaningful range, $-1 \leq \theta \leq 1$. Having found θ, we solve for σ^2,

$$\sigma^2 = c/\theta \tag{24.19}$$

[[TODO: Numbers to R in this paragraph]]

The method of moments estimator takes the sample values of these moments, $\hat{v}$ and $\hat{c}$, and plugs them in to Eqs. 24.18 and 24.19. With the S& P returns, the sample covariance is -1.61×10^{-5}, and the sample variance 1.96×10^{-4}. This leads to $\hat{\theta}_{MM} = -8.28 \times 10^{-2}$, and $\widehat{\sigma^2}_{MM} = 1.95 \times 10^{-4}$. In terms of the model, then, each day's news has a follow-on impact on prices which is about 8% as large as its impact the first day, but with the opposite sign.[7]

If we did not know how to solve a quadratic equation, we could use the minimization version of the method of moments estimator:

$$\begin{bmatrix} \widehat{\theta}_{MM} \\ \widehat{\sigma^2_{MM}} \end{bmatrix} = \underset{\theta,\sigma^2}{\mathrm{argmin}} \left\| \begin{array}{c} \sigma^2\theta - \hat{c} \\ \sigma^2(1+\theta^2) - \hat{v} \end{array} \right\|^2 \tag{24.20}$$

turns out that models which are not "invertible" in this way are very strange — see Shumway and Stoffer (2000).

[6] For example, plot c/v as a function of θ, and observe that any horizontal line cuts the graph at only one point.

[7] It would be natural to wonder whether $\widehat{\theta_{MM}}$ is really significantly different from zero. Assuming Gaussian noise, one could, in principle, calculate the probability that even though $\theta = 0$, by chance $\hat{c}/\hat{v}$ was so far from zero as to give us our estimate. As you will see in the homework, however, Gaussian assumptions are very bad for this data. This sort of thing is why we have bootstrapping.

```
ma.mm.est <- function(c, v) {
    theta.0 <- c/v
    sigma2.0 <- v
    fit <- optim(par = c(theta.0, sigma2.0), fn = ma.mm.objective, c = c, v = v)
    return(fit)
}


ma.mm.objective <- function(params, c, v) {
    theta <- params[1]
    sigma2 <- params[2]
    c.pred <- theta * sigma2
    v.pred <- sigma2 * (1 + theta^2)
    return((c - c.pred)^2 + (v - v.pred)^2)
}
```

CODE EXAMPLE 35: *Code for implementing method of moments estimation of a first-order moving average model, as in Eq. 24.20. See App. J.9.7 for "design notes", and the online code for comments.*

```
rma <- function(n, theta, sigma2, s = 1) {
    z <- replicate(s, rnorm(n = n + 1, mean = 0, sd = sqrt(sigma2)))
    x <- z[-1, ] + theta * z[-(n + 1), ]
    return(x)
}
```

CODE EXAMPLE 36: *Function which simulates* `s` *independent runs of a first-order moving average model, each of length* `n`*, with given noise variance* `sigma2` *and after-effect* `theta`*. See online for comments.*

Computationally, it would go something like Code Example 35.

The parameters estimated by minimization agree with those from direct algebra to four significant figures, which I hope is good enough to reassure you that this works.

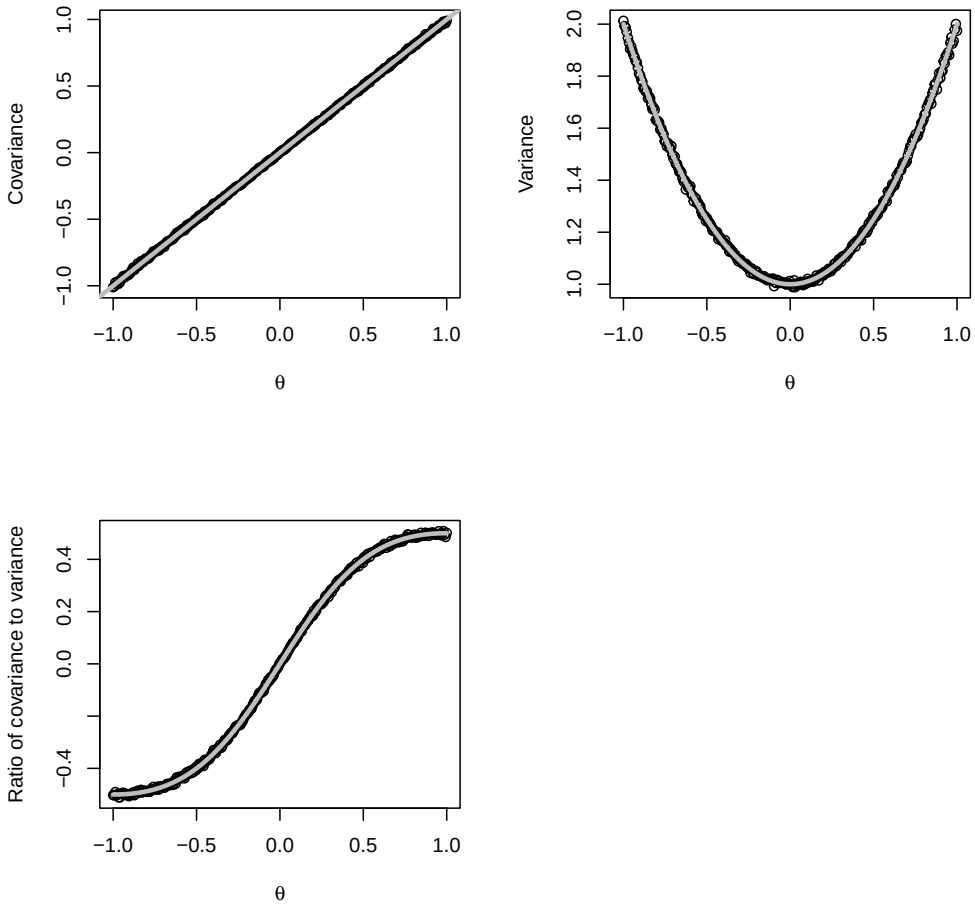
Before we can try out the method of simulated moments, we have to figure out how to simulate our model. X_t is a deterministic function of Z_t and Z_{t-1}, so our general strategy (§5.2.1) says to first generate the Z_t, and then compute X_t from that. But here the Z_t are just a sequence of independent Gaussians, which is a solved problem for us. The one wrinkle is that to get our first value X_1, we need a previous value Z_0. Code Example 36 shows the solution. v

What we need to extract from the simulation are the variance and the covariance. It will be more convenient to have functions which calculate these call `rma()` themselves (Code Example 37).

Figure 24.1 plots the covariance, the variance, and their ratio as functions of θ with $\sigma^2 = 1$, showing both the values obtained from simulation and the theoretical ones.[8] The agreement is quite good, though of course not quite perfect.[9]

[8] I could also have varied σ^2 and made 3D plots, but that would have been more work. Also, the variance and covariance are both proportional to σ^2, so the shapes of the figures would all be the same.

[9] If you look at those figures and think "Why not do a nonparametric regression of the simulated




```
par(mfrow = c(2, 2))
theta.grid <- seq(from = -1, to = 1, length.out = 300)
cov.grid <- sapply(theta.grid, sim.cov, sigma2 = 1, n = length(sp), s = 10)
plot(theta.grid, cov.grid, xlab = expression(theta), ylab = "Covariance")
abline(0, 1, col = "grey", lwd = 3)
var.grid <- sapply(theta.grid, sim.var, sigma2 = 1, n = length(sp), s = 10)
plot(theta.grid, var.grid, xlab = expression(theta), ylab = "Variance")
curve((1 + x^2), col = "grey", lwd = 3, add = TRUE)
plot(theta.grid, cov.grid/var.grid, xlab = expression(theta), ylab = "Ratio of covariance to varianc
curve(x/(1 + x^2), col = "grey", lwd = 3, add = TRUE)
par(mfrow = c(1, 1))
```

Figure 24.1 Plots of the covariance, the variance, and their ratio as a function of θ, with $\sigma^2 = 1$. Dots show simulation values (averaging 10 realizations each as long as the data), the grey curves the exact calculations.

```
sim.var <- function(n, theta, sigma2, s = 1) {
    vars <- apply(rma(n, theta, sigma2, s), 2, var)
    return(mean(vars))
}

sim.cov <- function(n, theta, sigma2, s = 1) {
    x <- rma(n, theta, sigma2, s)
    covs <- colMeans(x[-1, ] * x[-n, ])
    return(mean(covs))
}
```

CODE EXAMPLE 37: *Functions for calculating the variance and covariance for specified parameter values from simulations.*

```
ma.msm.est <- function(c, v, n, s) {
    theta.0 <- c/v
    sigma2.0 <- v
    fit <- optim(par = c(theta.0, sigma2.0), fn = ma.msm.objective, c = c, v = v,
        n = n, s = s)
    return(fit)
}

ma.msm.objective <- function(params, c, v, n, s) {
    theta <- params[1]
    sigma2 <- params[2]
    c.pred <- sim.cov(n, theta, sigma2, s)
    v.pred <- sim.var(n, theta, sigma2, s)
    return((c - c.pred)^2 + (v - v.pred)^2)
}
```

CODE EXAMPLE 38: *Code for implementing the method of simulated moments estimation of a first-order moving average model.*

Conceptually, we could estimate θ by jut taking the observed value $\hat{c}/\hat{v}$, running a horizontal line across Figure 24.1c, and seeing at what θ it hit one of the simulation dots. Of course, there might not be one it hits *exactly...*

The more practical approach is Code Example 38. The code is practically identical to that in Code Example 35, except that the variance and covariance predicted by given parameter settings now come from simulating those settings, not an exact calculation. Also, we have to say how long a simulation to run, and how many simulations to average over per parameter value.

When I run this, with `s=100`, I get $\widehat{\theta}_{MSM} = -8.36 \times 10^{-2}$ and $\widehat{\sigma}^2_{MSM} = 1.94 \times 10^{-4}$, which is quite close to the non-simulated method of moments estimate. In fact, in this case there is actually a maximum likelihood estimator (`arima()`, after the more general class of models including MA models), which claims $\widehat{\theta}_{ML} = -9.75 \times 10^{-2}$ and $\widehat{\sigma}^2_{ML} = 1.94 \times 10^{-4}$. Since the standard error of the MLE on θ is ± 0.02, this is working essentially as well as the method of moments, or even the method of simulated moments.

[[TODO: Replace numbers with R]]

moments against the parameters and use the fitted values as $\tilde{g}$, it'll get rid of some of the simulation noise?", congratulations, you've just discovered the smoothed method of simulated moments.

In this case, because there *is* a tractable maximum likelihood estimator, one generally wouldn't use the method of simulated moments. But we can in this case check whether it works (it does), and so we can use the same technique for other models, where an MLE is unavailable.

24.2 Indirect Inference

Section 24.1 explained the method of simulated moments, where we try to match expectations of various functions of the data. Expectations of functions are summary statistics, but they're not the *only* kind of summary statistics. We could try to estimate our model by matching any set of summary statistics, so long as (i) there's a unique way of mapping back from summaries to parameters, and (ii) estimates of the summary statistics converge as we get more data.

A powerful but somewhat paradoxical version of this is what's called **indirect inference**, where the summary statistics are the parameters of a *different* model. This second or **auxiliary** model does *not* have to be correctly specified, it just has to be easily fit to the data, and satisfy (i) and (ii) above. Say the parameters of the auxiliary model are β, as opposed to the θ of our real model. We calculate $\widehat{\beta}$ on the real data. Then we simulate from different values of θ, fit the auxiliary to the simulation outputs, and try to match the auxiliary estimates. Specifically, the indirect inference estimator is

$$\widehat{\theta}_{II} = \operatorname*{argmin}_{\theta} \|\tilde{\beta}(\theta) - \widehat{\beta}\|^2 \tag{24.21}$$

where $\tilde{\beta}(\theta)$ is the value of β we estimate from a simulation of θ, of the same size as the original data. (We might average together a couple of simulation runs for each θ.) If we have a consistent estimator of β, then

$$\widehat{\beta} \to \beta \tag{24.22}$$

$$\tilde{\beta}(\theta) \to b(\theta) \tag{24.23}$$

If in addition $b(\theta)$ is invertible, then

$$\widehat{\theta}_{II} \to \theta \tag{24.24}$$

For this to work, the auxiliary model needs to have at least as many parameters as the real model, but we can often arrange this by, say, making the auxiliary model a linear regression with a lot of coefficients.

[[TODO: nominal confidence limits by means of the information matrix]]

A specific case, often useful for time series, is to make the auxiliary model a linear **autoregressive model** (§23.4), where each observation is linearly regressed on the previous ones — see the discussion in §23.4.

24.3 Further Reading

The best general reference on simulation-based inference I know is (despite its age) still Gouriéroux and Monfort (1989/1995); many of the examples presume

some familiarity with the jargon of econometrics, but the general approaches do not. It covers the simulated method of moments, simulated maximum likelihood, and (unsurprisingly: Gouriéroux *et al.* 1993) indirect inference.

Kendall *et al.* (2005) is an excellent example of applying indirect inference to testing substantive scientific (not statistical!) hypotheses with real data. (I learned about indirect inference from hearing Prof. Ellner describe this paper.)

The weakest conditions I know of under which indirect inference is consistent are given in Zhao (2010, ch. 5).

Wood (2010) proposes an interesting variant on indirect inference; despite the title, it applies much more generally than to ecology.

Indirect inference has a Bayesian counterpart, or even version, called "approximate Bayesian computation", which originated in population genetics; Beaumont (2010) is an accessible review by one of the inventors.

Exercises

24.1 *Indirect inference*

1. Convince yourself that if X_t comes from an MA(1) process, it can't also be written as an AR(1) model.
2. Write a function, `ar1.fit`, to fit an AR(1) model to a time series, using `lm`, and to return the three parameters (intercept, slope, noise variance).
3. Apply `ar1.fit` to the S&P 500 data; what are the auxiliary parameter estimates?
4. Combine `ar1.fit` with the simulator `rma`, and plot the three auxiliary parameters as functions of θ, holding σ^2 fixed at 1. (This is analogous to Figure 24.1.)
5. Write functions, analogous to `ma.msm.est` and `ma.msm.objective`, for estimating an MA(1) model, using an AR(1) model as the auxiliary function. Does this recover the right parameter values when given data simulated from an MA(1) model?
6. What values does your estimator give for θ and σ^2 on the S& P 500 data? How do they compare to the other estimates?

24.2 *Indirect inference with a mechanistic model* [[TODO: Lotka-Volterra model with errors-in-variables for the lynx data?]]

References

Abadie, Alberto and Guido W. Imbens (2006). "Large Sample Properties of Matching Estimators for Average Treatment Effects." *Econometrica*, **74**: 235–267. URL `http://www.ksg.harvard.edu/fs/aabadie/smep.pdf`. doi:10.1111/j.1468-0262.2006.00655.x. 495

Abarbanel, Henry D. I. (1996). *Analysis of Observed Chaotic Data*. Berlin: Springer-Verlag. 550

Aczél, J. and Zoltán Daróczy (1975). *On Measures of Information and Their Characterizations*. New York: Academic Press. 663

Adler, Joseph (2009). *R in a Nutshell*. Sebastopol, California: O'Reilly. 796

Akaike, Hirotugu (1973). "Information Theory and an Extension of the Maximum Likelihood Principle." In *Proceedings of the Scond International Symposium on Information Theory* (B. N. Petrov and F. Caski, eds.), pp. 267–281. Budapest: Akademiai Kiado. Reprinted in (Akaike, 1998, pp. 199–213). 660

— (1998). *Selected Papers of Hirotugu Akaike*. Berlin: Springer-Verlag. Edited by Emanuel Parzen, Kunio Tanabe and Genshiro Kitagawa. 583

al Ghazali, Abu Hamid Muhammad ibn Muhammad at-Tusi (1100/1997). *The Incoherence of the Philosophers = Tahafut al-Falasifah: A Parallel English-Arabic Text*. Provo, Utah: Brigham Young University Press. Translated by Michael E. Marmura. 456

Alford, J. R., C. L. Funk and J. R. Hibbibng (2005). "Are Political Orientations Genetically Transmitted?" *American Political Science Review*, **99**: 153–167. 239

Amari, Shun-ichi and Hiroshi Nagaoka (1993/2000). *Methods of Information Geometry*. Providence, Rhode Island: American Mathematical Society. Translated by Daishi Harada. As *Joho Kika no Hoho*, Tokyo: Iwanami Shoten Publishers. 712

Anthony, Martin and Peter L. Bartlett (1999). *Neural Network Learning: Theoretical Foundations*. Cambridge, England: Cambridge University Press. 93

Arceneaux, Kevin, Alan S. Gerber and Donald P. Green (2010). "A Cautionary Note on the Use of Matching to Estimate Causal Effects: An Empirical Example Comparing Matching Estimates to an Experimental Benchmark." *Sociological Methods and Research*, **39**: 256–282. doi:10.1177/0049124110378098. 495

Arlot, Sylvain and Alain Celisse (2010). "A survey of cross-validation procedures for model selection." *Statistics Surveys*, **4**: 40–79. URL `http://projecteuclid.org/euclid.ssu/1268143839`. doi:10.1214/09-SS054. 93

— (2011). "Segmentation of the mean of heteroscedastic data via cross-validation." *Statistics and Computing*, **21**: 613–632. URL `http://arxiv.org/abs/0902.3977`. 540

Arnold, Barry C. (1983). *Pareto Distributions*. Fairland, Maryland: International Cooperative Publishing House. 153, 420

Arnol'd, V. I. (1973). *Ordinary Differential Equations*. Cambridge, Massachusetts: MIT Press. Translated by Richard A. Silverman from *Obyknovennye differentsial'nye Uravneniya*. 712

Bai, Jushan (2003). "Testing Parametric Conditional Distributions of Dynamic Models." *The Review of Economics and Statistics*, **85**: 531–549. doi:10.1162/003465303322369704. 694

Barndorff-Nielsen, O. E. (1978). *Information and Exponential Families in Statistical Theory*. New York: John Wiley and Sons. 295

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

Barndorff-Nielsen, O. E. and D. R. Cox (1995). *Inference and Asymptotics*. London: Chapman and Hall. 660

Bartholomew, David J. (1987). *Latent Variable Models and Factor Analysis*. New York: Oxford University Press. 384, 395, 401, 405

Bartholomew, David J., Ian J. Deary and Martin Lawn (2009). "A New Lease on Life for Thomson's Bonds Model of Intelligence." *Psychological Review*, **116**: 567–579. doi:10.1037/a0016262. 399

Bartlett, M. S. (1955). *An Introduction to Stochastic Processes, with Special Reference to Methods and Applications*. Cambridge, England: Cambridge University Press. 549

Basharin, Gely P., Amy N. Langville and Valeriy A. Naumov (2004). "The Life and Work of A. A. Markov." *Linear Algebra and its Applications*, **386**: 3–26. URL `http://langvillea.people.cofc.edu/MarkovReprint.pdf`. 434

Baum, Leonard E., Ted Petrie, George Soules and Norman Weiss (1970). "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains." *Annals of Mathematical Statistics*, **41**: 164–171. URL `https://projecteuclid.org/euclid.aoms/1177697196`. doi:10.1214/aoms/1177697196. 548

Beaumont, Mark A. (2010). "Approximate Bayesian Computation in Evolution and Ecology." *Annual Review of Ecology, Evolution, and Systematics*, **41**: 379–406. doi:10.1146/annurev-ecolsys-102209-144621. 582

Becker, Howard S. (2017). *Evidence*. Chicago: University of Chicago Press. 755

Belkin, Mikhail and Partha Niyogi (2003). "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation." *Neural Computation*, **15**: 1373–1396. URL `http://www.cse.ohio-state.edu/~mbelkin/papers/LEM_NC_03.pdf`. doi:10.1162/089976603321780317. 728, 729

Benaglia, Tatiana, Didier Chauveau, David R. Hunter and Derek S. Young (2009). "mixtools: An R Package for Analyzing Mixture Models." *Journal of Statistical Software*, **32**. URL `http://www.jstatsoft.org/v32/i06`. 412, 606

Benzi, Michele (2009). "The Early History of Matrix Iterations: With a Focus on the Italian Contribution." Presentation at SIAM Conference on Applied Linear Algebra, Monterey Bay, California, 26 October 2009. URL `https://archive.siam.org/meetings/la09/talks/benzi.pdf`. 207

Bera, Anil K. and Aurobindo Ghosh (2002). "Neyman's Smooth Test and Its Applications in Econometrics." In *Handbook of Applied Econometrics and Statistical Inference* (Aman Ullah and Alan T. K. Wan and Anoop Chaturvedi, eds.), pp. 177–230. New York: Marcel Dekker. URL `http://ssrn.com/abstract=272888`. 684, 704

Berk, Richard, Lawrence Brown, Andreas Buja, Kai Zhang and Linda Zhao (2013). "Valid Post-Selection Inference." *Annals of Statistics*, **41**: 802–837. URL `http://arxiv.org/abs/1306.1059`. doi:10.1214/12-AOS1077. 93

Berk, Richard A. (2004). *Regression Analysis: A Constructive Critique*. Thousand Oaks, California: Sage. 66, 483

— (2008). *Statistical Learning from a Regression Perspective*. New York: Springer-Verlag. 315

Berk, Robert H. (1966). "Limiting Behavior of Posterior Distributions when the Model is Incorrect." *Annals of Mathematical Statistics*, **37**: 51–58. URL `http://projecteuclid.org/euclid.aoms/1177699597`. doi:10.1214/aoms/1177699597. See also correction, volume 37 (1966), pp. 745–746. 656

Beygelzimer, Alina, Sham Kakade, John Langford, Sunil Arya, David Mount and Shengqiao Li (2013). *FNN: Fast Nearest Neighbor Search Algorithms and Applications*. URL `http://CRAN.R-project.org/package=FNN`. R package version 1.1. 33, 606

Biecek, Przemyslaw and Teresa Ledwina (2010). *ddst: Data driven smooth test*. URL `http://CRAN.R-project.org/package=ddst`. R package, version 1.02. 690

Blackwell, David and M. A. Girshick (1954). *Theory of Games and Statistical Decisions*. New York: Wiley. 314

Blei, David M. and John D. Lafferty (2009). "Topic Models." In *Text Mining: Theory and Applications* (A. Srivastava and M. Sahami, eds.). London: Taylor and Francis. URL http://www.cs.columbia.edu/~blei/papers/BleiLafferty2009.pdf. 412

Blei, David M., Andrew Y. Ng and Michael I. Jordan (2003). "Latent Dirichlet Allocation." *Journal of Machine Learning Research*, **3**: 993–1022. URL http://jmlr.csail.mit.edu/papers/v3/blei03a.html. 412

Boas, Mary L. (1983). *Mathematical Methods in the Physical Sciences*. New York: Wiley, 2nd edn. 628

Bonner, John Tyler (1988). *The Evolution of Complexity, by Means of Natural Selection*. Princeton, New Jersey: Princeton University Press. 18, 57

Borsboom, Denny (2005). *Measuring the Mind: Conceptual Issues in Contemporary Psychometrics*. Cambridge, England: Cambridge University Press. 401

— (2006). "The Attack of the Psychometricians." *Psychometrika*, **71**: 425–440. URL https://sites.google.com/site/borsboomdenny/BorsboomPM2006.pdf. doi:10.1007/s11336-006-1447-6. 401

Boudon, Raymond (1998). "Social Mechanisms without Black Boxes." In Hedström and Swedberg (1998), pp. 172–203. doi:10.1017/CBO9780511663901. 483

Bousquet, Olivier, Stéphane Boucheron and Gábor Lugosi (2004). "Introduction to Statistical Learning Theory." In *Advanced Lectures in Machine Learning* (Olivier Bousquet and Ulrike von Luxburg and Gunnar Rätsch, eds.), pp. 169–207. Berlin: Springer-Verlag. URL http://www.econ.upf.edu/~lugosi/mlss_slt.pdf. 93

Boyd, Stephen and Lieven Vandenberghe (2004). *Convex Optimization*. Cambridge, England: Cambridge University Press. URL https://web.stanford.edu/~boyd/cvxbook/. 660, 677

Braun, W. John and Duncan J. Murdoch (2008). *A First Course in Statistical Programming with R*. Cambridge University Press. 796

Breiman, Leo, Jerome Friedman, R. Olshen and C. Stone (1984). *Classification and Regression Trees*. Belmont, California: Wadsworth. 315

Breiman, Leo and Jerome H. Friedman (1985). "Estimating Optimal Transformations for Multiple Regression and Correlation." *Journal of the American Statistical Association*, **80**: 580–598. doi:10.1080/01621459.1985.10478157. 60

Brown, Lawrence D. (1986). *Fundamentals of Statistical Exponential Families: with Applications in Statistical Decision Theory*. Hayward, California: Institute of Mathematical Statistics. URL http://projecteuclid.org/euclid.lnms/1215466757. 295

Bühlmann, Peter (2000). "Model Selection for Variable Length Markov Chains and Tuning the Context Algorithm." *Annals of the Institute of Statistical Mathematics*, **52**: 287–315. URL http://e-collection.ethbib.ethz.ch/show?type=incoll&nr=117. doi:10.1023/A:1004165822461. 526

— (2002). "Bootstraps for Time Series." *Statistical Science*, **17**: 52–72. URL http://projecteuclid.org/euclid.ss/1023798998. doi:10.1214/ss/1023798998. 549

Bühlmann, Peter and Abraham J. Wyner (1999). "Variable Length Markov Chains." *Annals of Statistics*, **27**: 480–513. URL http://projecteuclid.org/euclid.aos/1018031204. doi:10.1214/aos/1018031204. 526

Buja, Andreas, Richard Berk, Lawrence Brown, Edward George, Emil Pitkin, Mikhail Traskin, Linda Zhao and Kai Zhang (2014). "Models as Approximations, Part I: A Conspiracy of Nonlinearity and Random Regressors in Linear Regression." arxiv:1404.1578. URL http://arxiv.org/abs/1404.1578. 66, 167, 652

Buja, Andreas, Trevor Hastie and Robert Tibshirani (1989). "Linear Smoothers and Additive Models." *Annals of Statistics*, **17**: 453–555. URL http://projecteuclid.org/euclid.aos/1176347115. doi:10.1214/aos/1176347115. 43, 206

Burman, Prabir, Edmond Chow and Deborah Nolan (1994). "A Cross-Validatory Method for Dependent Data." *Biometrika*, **81**: 351–358. doi:10.1093/biomet/81.2.351. 549

Canty, Angelo and Brian Ripley (2013). "boot: Bootstrap R (S-Plus) Functions." R package version 1.3-9. URL http://cran.r-project.org. 167

Canty, Angelo J., Anthony C. Davison, David V. Hinkley and Valérie Ventura (2006). "Bootstrap Diagnostics and Remedies." *The Canadian Journal of Statistics*, **34**: 5–27. URL `http://www.stat.cmu.edu/tr/tr726/tr726.html`. doi:10.1002/cjs.5550340103. 167

Carmack, Patrick S., William R. Schucany, Jeffrey S. Spence, Richard F. Gunst, Qihua Lin and Robert W. Haley (2009). "Far Casting Cross Validation." *Journal of Computational and Graphical Statistics*, **18**: 879–893. doi:10.1198/jcgs.2009.07034. 549

Carroll, Raymond J., Aurore Delaigle and Peter Hall (2009). "Nonparametric Prediction in Measurement Error Models." *Journal of the American Statistical Association*, **104**: 993–1003. doi:10.1198/jasa.2009.tm07543. 57

Casella, George and R. L. Berger (2002). *Statistical Inference*. Belmont, California: Duxbury Press, 2nd edn. 295

Cavalli-Sforza, Luigi L., Paolo Menozzi and Alberto Piazza (1994). *The History and Geography of Human Genes*. Princeton: Princeton University Press. 366, 370

Chambers, John M. (2008). *Software for Data Analysis: Programming with R*. New York: Springer. 796

Chenoweth, Erica and Maria J. Stephan (2011). *Why Civil Resistance Works: The Strategic Logic of Nonviolent Conflict*. New York: Columbia University Press. 25

Chetty, Raj, Nathaniel Hendren, Patrick Kline and Emmanuel Saez (2014). "Where is the Land of Opportunity? The Geography of Intergenerational Mobility in the United States." *Quarterly Journal of Economics*, **129**: 1553–1623. URL `https://opportunityinsights.org/paper/land-of-opportunity/`. doi:10.1093/qje/qju022. 4, 15

Christakis, Nicholas A. and James H. Fowler (2007). "The Spread of Obesity in a Large Social Network over 32 Years." *The New England Journal of Medicine*, **357**: 370–379. URL `http://content.nejm.org/cgi/content/abstract/357/4/370`. 481

Chu, Tianjiao and Clark Glymour (2008). "Search for Additive Nonlinear Time Series Causal Models." *Journal of Machine Learning Research*, **9**: 967–991. URL `http://jmlr.csail.mit.edu/papers/v9/chu08a.html`. 506

Chu, Tianjiao, Clark Glymour, Richard Scheines and Peter Spirtes (2003). "A Statistical Problem for Inference to Regulatory Structure from Associations of Gene Expression Measurements with Microarrays." *Bioinformatics*, **19**: 1147–1152. 535

Claeskens, Gerda and Nils Lid Hjort (2008). *Model Selection and Model Averaging*. Cambridge, England: Cambridge University Press. doi:10.1017/CBO9780511790485. 93, 658, 659, 660, 687

Clauset, Aaron and Douglas H. Erwin (2008). "The Evolution and Distribution of Species Body Size." *Science*, **321**: 399–401. URL `http://arxiv.org/abs/0901.0251`. doi:10.1126/science.1157534. 18

Clauset, Aaron, Cosma Rohilla Shalizi and M. E. J. Newman (2009). "Power-law Distributions in Empirical Data." *SIAM Review*, **51**: 661–703. URL `http://arxiv.org/abs/0706.1062`. doi:10.1137/070710111. 151, 152

Cleveland, W. S. (1979). "Robust Locally Weighted Regression and Smoothing Scatterplots." *Journal of the American Statistical Association*, **74**: 829–836. URL `https://www.jstor.org/stable/2286407`. doi:10.2307/2286407. 256

Coleman, James, Elihu Katz and Herbert Menzel (1957). "The Diffusion of an Innovation Among Physicians." *Sociometry*, **20**: 253–270. URL `http://www.jstor.org/stable/2785979`. doi:10.2307/2785979. 96

Collier, Paul and Anke Hoeffler (2004). "Greed and Grievance in Civil War." *Oxford Economic Papers*, **56**: 563–595. URL `http://economics.ouls.ox.ac.uk/12055/1/2002-01text.pdf`. 22

Colombo, Diego, Marloes H. Maathuis, Markus Kalisch and Thomas S. Richardson (2012). "Learning High-dimensional Directed Acyclic Graphs with Latent And Selection Variables." *Annals of Statistics*, **40**: 249–321. URL `http://arxiv.org/abs/1104.5617`. doi:10.1214/11-AOS940. 505

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest and Clifford Stein (2001). *Introduction to Algorithms*. Cambridge, Massachusetts: MIT Press, 2nd edn. 0

Cover, Thomas M. (1968a). "Estimation by the Nearest Neighbor Rule." *IEEE Transactions on Information Theory*, **14**: 50–55. URL `http://www-isl.stanford.edu/~cover/papers/transIT/0050cove.pdf`. 43

— (1968b). "Rates of Convergence for Nearest Neighbor Procedures." In *Proceedings of the Hawaii International Conference on Systems Sciences* (B. K. Kinariwala and F. F. Kuo, eds.), pp. 413–415. Honolulu: University of Hawaii Press. URL `http://www-isl.stanford.edu/~cover/papers/paper009.pdf`. 43

Cover, Thomas M. and P. E. Hart (1967). "Nearest Neighbor Pattern Classification." *IEEE Transactions on Information Theory*, **13**: 21–27. URL `http://www-isl.stanford.edu/~cover/papers/transIT/0021cove.pdf`. 43

Cover, Thomas M. and Joy A. Thomas (2006). *Elements of Information Theory*. New York: John Wiley, 2nd edn. 443, 450, 676

Cramér, Harald (1945). *Mathematical Methods of Statistics*. Uppsala: Almqvist and Wiksells. 660

Cristianini, Nello and John Shawe-Taylor (2000). *An Introduction to Support Vector Machines: And Other Kernel-Based Learning Methods*. Cambridge, England: Cambridge University Press. 93

Crutchfield, James P. and Karl Young (1989). "Inferring Statistical Complexity." *Physical Review Letters*, **63**: 105–108. URL `http://csc.ucdavis.edu/~cmg/compmech/pubs/ISCTitlePage.htm`. doi:10.1103/PhysRevLett.63.105. 550

Daley, D. J. and D. Vere-Jones (2003). *Elementary Theory and Methods*, vol. 1 of *An Introduction to the Theory of Point Processes*. New York: Springer-Verlag, 2nd edn. 550

Dasgupta, Sanjoy and Anupam Gupta (2002). "An Elementary Proof of a Theorem of Johnson and Lindenstrauss." *Random Structures and Algorithms*, **22**: 60–65. URL `http://cseweb.ucsd.edu/~dasgupta/papers/jl.pdf`. doi:10.1002/rsa.10073. 367

Davison, A. C. (2013). *SMPracticals: Practicals for use with Davison (2003)* Statistical Models. URL `https://CRAN.R-project.org/package=SMPracticals`. R package version 1.4-2. 606

Davison, A. C. and D. V. Hinkley (1997). *Bootstrap Methods and their Applications*. Cambridge, England: Cambridge University Press. doi:10.1017/CBO9780511802843. 167, 549

Dawes, Robyn M. (1975). "Graduate Admission Variables and Future Success." *Science*, **187**: 721–723. URL `https://www.jstor.org/stable/1739800`. doi:10.1126/science.187.4178.721. 742

de Oliveira, Cesar, Richard Watt and Mark Hamer (2010). "Toothbrushing, inflammation, and risk of cardiovascular disease: results from Scottish Health Survey." *British Medical Journal*, **340**: c2451. doi:10.1136/bmj.c2451. 459

Deaton, Angus (2010). "Instruments, Randomization, and Learning about Development." *Journal of Economic Literature*, **48**: 424–455. doi:10.1257/jel.48.2.424. 478, 483

Deerwester, Scott, Susan T. Dumais, George W. Furnas, Thomas K. Landauer and Richard Harshman (1990). "Indexing by Latent Semantic Analysis." *Journal of the American Society for Information Science*, **41**: 391–407. URL `http://lsa.colorado.edu/papers/JASIS.lsi.90.pdf`. doi:10.1002/(SICI)1097-4571(199009)41:6¡391::AID-ASI1¿3.0.CO;2-9. 362, 368

DeLanda, Manuel (2006). *A New Philosophy of Society: Assemblage Theory and Social Complexity*. London: Continuum. 483

Devroye, Luc (1986). *Non-Uniform Random Variate Generation*. Berlin: Springer-Verlag. URL `http://www.nrbook.com/devroye/`. 138

Devroye, Luc and Gábor Lugosi (2001). *Combinatorial Methods in Density Estimation*. Berlin: Springer-Verlag. 325, 344

Diaconis, Persi and David Freedman (1980). "De Finetti's Theorem for Markov Chains." *Annals of Probability*, **8**: 115–130. URL `http://projecteuclid.org/euclid.aop/1176994828`. doi:10.1214/aop/1176994828. 525

Didelez, Vanessa, Sha Meng and Nuala A. Sheehan (2010). "Assumptions of IV Methods for Observational Epidemiology." *Statistical Science*, **25**: 22–40. URL http://arxiv.org/abs/1011.0595. 483

Ding, Peng and Fang Li (2018). "Causal Inference: A Missing Data Perspective." *Statistical Science*, **33**: 214–237. URL https://projecteuclid.org/euclid.ss/1525313143. doi:10.1214/18-STS645. 747

Dinno, Alexis (2009). *LoopAnalyst: A collection of tools to conduct Levins' Loop Analysis*. URL http://CRAN.R-project.org/package=LoopAnalyst. R package version 1.2-2. 448

Douc, Randal, Eric Moulines and David S. Stoffer (2014). *Nonlinear Time Series: Theory, Methods, and Applications with R Examples*. Boca Raton, Florida: Chapman Hall/CRC. 549

Dreger, Alice (1998). *Hermaphrodites and the Medical Invention of Sex*. Cambridge, Massachusetts: Harvard University Press. 487

DuMouchel, William H. and Greg J. Duncan (1983). "Using Sample Survey Weights in Multiple Regression Analyses of Stratified Samples." *Journal of the American Statistical Association*, **78**: 535–543. URL http://www.jstor.org/stable/2288115. 226

Durbin, J. (1960). "The fitting of time series models." *Revue de l'Institut International de Statistique / Review of the International Statistical Institute*, **28**: 233–244. URL https://repository.lib.ncsu.edu/handle/1840.4/2230. doi:10.2307/1401322. 546

Durbin, James and Siem Jam Koopman (2001). *Time Series Analysis by State Space Methods*. Oxford: Oxford University Press. 544

Efron, Bradley (1979). "Bootstrap Methods: Another Look at the Jackknife." *Annals of Statistics*, **7**: 1–26. URL http://projecteuclid.org/euclid.aos/1176344552. doi:10.1214/aos/1176344552. 141, 167

— (1982). *The Jackknife, the Bootstrap, and Other Resampling Plans*. Philadelphia: SIAM Press. 167

— (1986). "How Biased is the Apparent Error Rate of a Prediction Rule?" *Journal of the American Statistical Association*, **81**: 461–470. URL https://www.jstor.org/stable/2289236. doi:10.2307/2289236. 43, 94

— (2004). "The Estimation of Prediction Error: Covariance Penalties and Cross-Validation." *Journal of the American Statistical Association*, **99**: 619–632. URL https://www.jstor.org/stable/27590436. doi:10.1198/016214504000000692. 43

Efron, Bradley and Robert J. Tibshirani (1993). *An Introduction to the Bootstrap*. New York: Chapman and Hall. 167

Eliade, Mircea (1971). *The Forge and the Crucible: The Origin and Structure of Alchemy*. New York: Harper and Row. 64

Elster, Jon (1989). *Nuts and Bolts for the Social Sciences*. Cambridge, England: Cambridge University Press. 483

Entner, Doris, Patrik O. Hoyer and Peter Spirtes (2013). "Data-driven Covariate Selection for Nonparametric Estimation of Causal Effects." In *Sixteenth International Conference on Artificial Intelligence and Statistics [AISTATS 2013]* (Carlos M. Carvalho and Pradeep Ravikumar, eds.), pp. 256–264. URL http://jmlr.org/proceedings/papers/v31/entner13a.html. 470

Eshel, Gidon (2012). *Spatiotemporal Data Analysis*. Princeton, New Jersey: Princeton University Press. doi:10.1515/9781400840632. 368

Ezekiel, Mordecai (1924). "A Method of Handling Curvilinear Correlation for Any Number of Variables." *Journal of the American Statistical Association*, **19**: 431–453. URL http://www.jstor.org/stable/2281561. 207

Fair, Ray C. (1978). "A Theory of Extramarital Affairs." *Journal of Political Economy*, **86**: 45–61. URL http://fairmodel.econ.yale.edu/rayfair/pdf/1978A200.PDF. 55

Fama, Eugene F. and Kenneth R. French (1993). "Common risk factors in the returns on stocks and bonds." *Journal of Financial Economics*, **33**: 3–56. doi:10.1016/0304-405X(93)90023-5. 30

Fan, Jianqing and Qiwei Yao (2003). *Nonlinear Time Series: Nonparametric and Parametric Methods*. Berlin: Springer-Verlag. 549

Fan, Jiaqing and I. Gijbels (1996). *Local Polynomial Modelling and Its Applications*. London: Chapman and Hall. 256

Faraway, Julian J. (1992). "On the Cost of Data Analysis." *Journal of Computational and Graphical Statistics*, **1**: 213–229. URL `http://people.bath.ac.uk/jjf23/papers/cda.pdf`. doi:10.1080/10618600.1992.10474582. 93

— (2004). *Linear Models with R*. Boca Raton, Florida: Chapman and Hall/CRC Press. 66

— (2006). *Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. Boca Raton, Florida: Chapman and Hall/CRC. 43, 296

— (2014). *faraway: Functions and datasets for books by Julian Faaway*. URL `https://CRAN.R-project.org/package=faraway`. R package version 1.0.6. 606

— (2016). "Does Data Splitting Improve Prediction?" *Statistics and Computing*, **26**: 49–60. URL `http://arxiv.org/abs/1301.2983`. doi:10.1007/s11222-014-9522-9. 90, 93

Farebrother, Richard William (1999). *Fitting Linear Relationships: A History of the Calculus of Observations 1750–1900*. New York: Springer-Verlag. doi:10.1007/978-1-4612-0545-6. 66

Fisher, Franklin M. (1983). *Disequilibrium Foundations of Equilibrium Economics*. Cambridge, England: Cambridge University Press. 448

— (2010). "The Stability of General Equilibrium — What Do We Know and Why Is It Important?" In *General Equilibrium Analysis: A Century after Walras* (Pascal Bridel, ed.), pp. 34–45. London: Routledge. URL `http://economics.mit.edu/files/6988`. 448

Fisher, R. A. (1922). "On the Mathematical Foundations of Theoretical Statistics." *Philosophical Transactions of the Royal Society A*, **222**: 309–368. URL `http://digital.library.adelaide.edu.au/dspace/handle/2440/15172`. 660

Fraser, Andrew M. (2008). *Hidden Markov Models and Dynamical Systems*. Philadelphia: SIAM Press. URL `http://www.siam.org/books/ot107/`. 422, 544

Freedman, David A. (1983). "A Note on Screening Regression Equations." *The American Statistician*, **37**: 152–155. doi:10.1080/00031305.1983.10482792. 94

Frisch, Uriel (1995). *Turbulence: The Legacy of A. N. Kolmogorov*. Cambridge, England: Cambridge University Press. 549

Galles, David and Judea Pearl (1997). "Axioms of Causal Relevance." *Artificial Intelligence*, **97**: 9–43. URL `http://nexus.cs.usfca.edu/~galles/research/relaxiom.ps`. 473

Gamow, George (1970). *My World-Line: An Informal Autobiography*. New York: Viking Press. Foreword by Stanislaw M. Ulam. 627

Gelman, Andrew (2003). "A Bayesian Formulation of Exploratory Data Analysis and Goodness-of-fit Testing." *International Statistical Review*, **71**: 369–382. URL `http://www.stat.columbia.edu/~gelman/research/published/isr.pdf`. doi:10.1111/j.1751-5823.2003.tb00203.x. 134

Gelman, Andrew and Iain Pardoe (2007). "Average predictive comparisons for models with nonlinearity, interactions, and variance components." *Sociological Methodology*, **37**: 23–51. URL `http://www.stat.columbia.edu/~gelman/research/published/ape17.pdf`. doi:10.1111/j.1467-9531.2007.00181.x. 117

Gelman, Andrew and Cosma Rohilla Shalizi (2013). "Philosophy and the Practice of Bayesian Statistics." *British Journal of Mathematical and Statistical Psychology*, **66**: 8–38. URL `http://arxiv.org/abs/1006.3868`. doi:10.1111/j.2044-8317.2011.02037.x. 134, 536

Genz, Alan, Frank Bretz, Tetsuhisa Miwa, Xuefei Mi, Friedrich Leisch, Fabian Scheipl and Torsten Hothorn (2016). *mvtnorm: Multivariate Normal and t Distributions*. URL `http://CRAN.R-project.org/package=mvtnorm`. R package version 1.0-5. 606

Gershenfeld, Neil (1999). *The Nature of Mathematical Modeling*. Cambridge, England: Cambridge University Press. 181

Geyer, Charles J. (2013). "Asymptotics of Maximum Likelihood without the LLN or CLT or Sample Size Going to Infinity." In *Advances in Modern Statistical Theory and Applications: A Festschrift in honor of Morris L. Eaton* (Galin Jones and Xiaotong Shen, eds.), pp. 1–24. Beachwood, Ohio: Institute of Mathematical Statistics. URL http://arxiv.org/abs/1206.4762. doi:10.1214/12-IMSCOLL1001. 658, 660

Gilbert, Paul and Ravi Varadhan (2015). *numDeriv: Accurate Numerical Derivatives*. URL https://CRAN.R-project.org/package=numDeriv. R package version 2014.2-1. 606

Glymour, Clark (1986). "Statistics and Metaphysics." *Journal of the American Statistical Association*, **81**: 964–966. URL http://www.hss.cmu.edu/philosophy/glymour/glymour1986.pdf. 487

— (1998). "What Went Wrong? Reflections on Science by Observation and *The Bell Curve*." *Philosophy of Science*, **65**: 1–32. URL https://www.cmu.edu/dietrich/philosophy/docs/glymour/glymour1998.pdf. doi:10.1086/392624. 401

— (2001). *The Mind's Arrows: Bayes Nets and Graphical Causal Models in Psychology*. Cambridge, Massachusetts: MIT Press. 450

Glymour, Clark and Madelyn R. Glymour (2014). "Race and Sex are Causes." *Epidemiology*, **25**: 488–490. doi:10.1097/EDE.0000000000000122. 487

Gnedenko, B. V. and A. N. Kolmogorov (1954). *Limit Distributions for Sums of Independent Random Variables*. Cambridge, Massachusetts: Addison-Wesley. Translated from the Russian and annotated by K. L. Chung, with an Appendix by J. L. Doob. 647

Godfrey, L. G. (1988). *Misspecification Tests in Econometrics; The Lagrange Multiplier Principle and Other Approaches*. Cambridge, England: Cambridge University Press. 224

Gouriéroux, Christian and Alain Monfort (1989/1995). *Statistics and Econometric Models*. Themes in Modern Econometrics. Cambridge, England: Cambridge University Press. Translated by Quang Vuong from *Statistique et modèles économétriques*, Paris: Économica. 581

Gouriéroux, Christian, Alain Monfort and E. Renault (1993). "Indirect Inference." *Journal of Applied Econometrics*, **8**: S85–S118. URL http://www.jstor.org/stable/2285076. 582

Gray, Robert M. (1988). *Probability, Random Processes, and Ergodic Properties*. New York: Springer-Verlag. URL http://ee.stanford.edu/~gray/arp.html. 549

— (1990). *Entropy and Information Theory*. New York: Springer-Verlag. URL http://ee.stanford.edu/~gray/it.html. 669

— (2009). *Probability, Random Processes, and Ergodic Properties*. New York: Springer-Verlag, 2nd edn. URL http://ee.stanford.edu/~gray/arp.html. 523

Gretton, Arthur, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf and Alexander Smola (2012). "A Kernel Two-Sample Test." *Journal of Machine Learning Research*, **13**: 723–773. URL http://jmlr.csail.mit.edu/papers/v13/gretton12a.html. 499

Griffeath, David (1976). "Introduction to Markov Random Fields." In *Denumerable Markov Chains* (John G. Kemeny and J. Laurie Snell and Anthony W. Knapp, eds.), pp. 425–457. Berlin: Springer-Verlag, 2nd edn. 447, 450

Grimmett, G. R. and D. R. Stirzaker (1992). *Probability and Random Processes*. Oxford: Oxford University Press, 2nd edn. 549

Grünwald, Peter D. (2007). *The Minimum Description Length Principle*. Cambridge, Massachusetts: MIT Press. 295

Guckenheimer, John and Philip Holmes (1983). *Nonlinear Oscillations, Dynamical Systems and Bifurcations of Vector Fields*. New York: Springer-Verlag. 712

Guttorp, Peter (1995). *Stochastic Modeling of Scientific Data*. London: Chapman and Hall. 137, 274, 283, 295, 450, 549, 550

Györfi, László, Michael Kohler, Adam Krzyżak and Harro Walk (2002). *A Distribution-Free Theory of Nonparametric Regression*. New York: Springer-Verlag. 34, 43, 93

Hacking, Ian (1990). *The Taming of Chance*. Cambridge, England: Cambridge University Press. doi:10.1017/CBO9780511819766. 19

— (2001). *An Introduction to Probability and Inductive Logic*. Cambridge, England: Cambridge University Press. 536

Hall, Peter and Joel Horowitz (2013). "A Simple Bootstrap Method for Constructing Nonparametric Confidence Bands for Functions." *Annals of Statistics*, **41**: 1892–1921. URL `https://projecteuclid.org/euclid.aos/1378386242`. doi:10.1214/13-AOS1137. 164

Hall, Peter, Jeff Racine and Qi Li (2004). "Cross-Validation and the Estimation of Conditional Probability Densities." *Journal of the American Statistical Association*, **99**: 1015–1026. URL `http://www.ssc.wisc.edu/~bhansen/workshop/QiLi.pdf`. 334, 343, 500

Halmos, Paul R. (1957). *An Introduction to Hilbert Space and the Theory of Spectral Multiplicity*. New York: Chelsea Publishing Co., 2nd edn. First edition, 1951. 207

Hand, David, Heikki Mannila and Padhraic Smyth (2001). *Principles of Data Mining*. Cambridge, Massachusetts: MIT Press. 368

Handcock, Mark S. (2015). *Relative Distribution Methods*. URL `http://CRAN.R-project.org/package=reldist`. Version 1.6-4. 606

Handcock, Mark S. and Martina Morris (1998). "Relative Distribution Methods." *Sociological Methodology*, **28**: 53–97. URL `http://www.jstor.org/stable/270964`. doi:10.1111/0081-1750.00042. 697, 704

— (1999). *Relative Distribution Methods in the Social Sciences*. Berlin: Springer-Verlag. doi:10.1007/b97852. 697, 704

Hannan, E. J. and Jorma Rissanen (1982). "Recursive estimation of mixed autoregressive-moving average order." *Biometrika*, **69**: 81–94. URL `https://www.jstor.org/stable/2335856`. doi:10.2307/2335856. 546

Hansen, Ben B. and Stephanie Olsen Klopfer (2006). "Optimal full matching and related designs via network flows." *Journal of Computational and Graphical Statistics*, **15**: 609–627. URL `http://www.stat.lsa.umich.edu/~bbh/hansenKlopfer2006.pdf`. doi:10.1198/106186006X137047. 496

Hart, Jeffrey D. (1997). *Nonparametric Smoothing and Lack-of-Fit Tests*. Berlin: Springer-Verlag. 210, 224

Hartley, R. V. L. (1928). "Transmission of Information." *Bell System Technical Journal*, pp. 535–563. 676

Hastie, Trevor and Robert Tibshirani (1990). *Generalized Additive Models*. London: Chapman and Hall. 43

Hastie, Trevor, Robert Tibshirani and Jerome Friedman (2009). *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Berlin: Springer, 2nd edn. URL `http://www-stat.stanford.edu/~tibs/ElemStatLearn/`. 66, 93, 182, 643

Hayfield, Tristen and Jeffrey S. Racine (2008). "Nonparametric Econometrics: The `np` Package." *Journal of Statistical Software*, **27(5)**: 1–32. URL `http://www.jstatsoft.org/v27/i05`. 102, 118, 606

Heckman, James J. (1976). "The Common Structure of Statistical Models of Truncation, Sample Selection and Limited Dependent Variables and a Simple Estimator for Such Models." *Annals of Economic and Social Measurement*, **5**: 475–492. URL `http://www.nber.org/chapters/c10491`. 745, 746

— (1979). "Sample Bias as a Specification Error." *Econometrica*, **47**: 153–162. URL `http://www.nber.org/papers/w0172`. 746

Hedström, Peter (2005). *Dissecting the Social: On the Principles of Analytical Sociology*. Cambridge, England: Cambridge University Press. 483

Hedström, Peter and Richard Swedberg (eds.) (1998). *Social Mechanisms: An Analytical Approach to Social Theory*, Cambridge, England. Cambridge University Press. doi:10.1017/CBO9780511663901. 483, 585

Ho, Daniel E., Kosuke Imai, Gary King and Elizabeth A. Stuart (2011). "MatchIt: Nonparametric Preprocessing for Parametric Causal Inference." *Journal of Statistical Software*, **42(8)**: 1–28. URL `http://www.jstatsoft.org/v42/i08/`. 496

Hoerl, Arthur E. and Robert W. Kennard (1970). "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics*, **12**. URL `http://www.jstor.org/pss/1267351`. doi:10.2307/1267351. 642

Hofmann, Thomas (1999). "Probabilistic Latent Semantic Analysis." In *Uncertainty in Artificial Intelligence: Proceedings of the Fifteenth Conference [UAI 1999]* (Kathryn Laskey and Henri Prade, eds.), pp. 289–296. San Francisco: Morgan Kaufmann. URL http://uai.sis.pitt.edu/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=179&proceeding_id=15. 412

Holan, Scott H., Robert Lund and Ginger Davis (2010). "The ARMA alphabet soup: A tour of ARMA model variants." *Statistics Surveys*, **4**: 232–274. URL http://projecteuclid.org/euclid.ssu/1291731822. doi:10.1214/09-SS060. 549

Holland, Paul W. (1986). "Statistics and Causal Inference." *Journal of the American Statistical Association*, **81**: 945–970. 487

Honerkamp, Josef (2002). *Statistical Physics: An Advanced Approach with Applications*. Berlin: Springer-Verlag, 2nd edn. Translated by Thomas Filk. 450

Hong, Yongmiao and Halbert White (1995). "Consistent Specification Testing Via Nonparametric Series Regression." *Econometrica*, **63**: 1133–1159. URL http://www.jstor.org/stable/2171724. doi:10.2307/2171724. 224

Hotelling, Harold (1933a). "Analysis of a complex of statistical variables into principal components [Part 1 of 2]." *Journal of Educational Psychology*, **24**: 471–441. doi:10.1037/h0071325. 368

— (1933b). "Analysis of a complex of statistical variables into principal components [Part 2 of 2]." *Journal of Educational Psychology*, **24**: 498–520. doi:10.1037/h0070888. 368

Hoyer, Patrik O., Domink Janzing, Joris Mooij, Jonas Peters and Bernhard Schölkopf (2009). "Nonlinear causal discovery with additive noise models." In *Advances in Neural Information Processing Systems 21 [NIPS 2008]* (Daphne Koller and D. Schuurmans and Y. Bengio and Léon Bottou, eds.), pp. 689–696. Cambridge, Massachusetts: MIT Press. URL http://books.nips.cc/papers/files/nips21/NIPS2008_0266.pdf. 502

Huber, Peter J. (1967). "The Behavior of Maximum Likelihood Estimates under Nonstandard Conditions." In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (Lucien M. Le Cam and Jerzy Neyman, eds.), vol. 1, pp. 221–233. Berkeley: University of California Press. URL http://projecteuclid.org/euclid.bsmsp/1200512988. 652, 660

Hume, David (1739). *A Treatise of Human Nature: Being an Attempt to Introduce the Experimental Method of Reasoning into Moral Subjects*. London: John Noon. Reprint (Oxford: Clarendon Press, 1951) of original edition, with notes and analytical index. 456

Hunter, David R., Steven M. Goodreau and Mark S. Handcock (2008). "Goodness of Fit of Social Network Models." *Journal of the American Statistical Association*, **103**: 248–258. URL http://www.csss.washington.edu/Papers/wp47.pdf. doi:10.1198/016214507000000446. 134

Imbens, Guido W. and Donald B. Rubin (2015). *Causal Inference for Statistics, Social, and Biomedical Sciences: An Introduction*. Cambridge, England: Cambridge University Press. 461, 496, 600

Iyigun, Murat (2008). "Luther and Suleyman." *Quarterly Journal of Economics*, **123**: 1465–1494. URL http://www.colorado.edu/Economics/courses/iyigun/ottoman081506.pdf. doi:10.1162/qjec.2008.123.4.1465. 91

Jacobs, Robert A. (1997). "Bias/Variance Analyses of Mixtures-of-Experts Architectures." *Neural Computation*, **9**: 369–383. 422

Jaeger, Herbert (2000). "Observable Operator Models for Discrete Stochastic Time Series." *Neural Computation*, **12**: 1371–1398. URL http://minds.jacobs-university.de/sites/default/files/uploads/papers/oom_neco00.pdf. doi:10.1162/089976600300015411. 550

Janzing, Dominik (2007). "On causally asymmetric versions of Occam's Razor and their relation to thermodynamics." arxiv:0708.3411. URL http://arxiv.org/abs/0708.3411. 459, 502, 503

Janzing, Dominik and Daniel Herrmann (2003). "Reliable and Efficient Inference of Bayesian Networks from Sparse Data by Statistical Learning Theory." arxiv:cs.LG/0309015. URL http://arxiv.org/abs/cs.LG/0309015. 512

Jaynes, E. T. (1983). *Essays on Probability, Statistics, and Statistical Physics*. Dordrecht: D. Reidel. Edited by R. D. Rosenkrantz. 677

— (2003). *Probability Theory: The Logic of Science*. Cambridge, England: Cambridge University Press. 677

Jones, Owen, Robert Maillardet and Andrew Robinson (2009). *Introduction to Scientific Programming and Simulation Using R*. Boca Raton, Florida: Chapman and Hall/CRC. 137

Jordan, Michael I. (ed.) (1998). *Learning in Graphical Models*, Dordrecht. Kluwer Academic. 446, 450, 597

Jordan, Michael I. (2013). "On statistics, computation and scalability." *Bernoulli*, **19**: 1378–1390. doi:http://dx.doi.org/10.3150/12-BEJSP17. 660

Jordan, Michael I. and Robert A. Jacobs (1994). "Hierarchical Mixtures of Experts and the EM Algorithm." *Neural Computation*, **6**: 181–214. URL `ftp://publications.ai.mit.edu/ai-publications/pdf/AIM-1440.pdf`. doi:10.1162/neco.1994.6.2.181. 422

Jordan, Michael I. and Terrence J. Sejnowski (eds.) (2001). *Graphical Models: Foundations of Neural Computation*, Computational Neuroscience, Cambridge, Massachusetts. MIT Press. 450

Kahn, Joan R. and J. Richard Udry (1986). "Marital Coital Frequency: Unnoticed Outliers and Unspecified Interactions Lead to Erroneous Conclusions." *American Sociological Review*, **51**: 734–737. URL `https://www.jstor.org/stable/2095496`. doi:10.2307/2095496. 735

Kalisch, Markus and Peter Bühlmann (2007). "Estimating High-Dimensional Directed Acyclic Graphs with the PC-Algorithm." *Journal of Machine Learning Research*, **8**: 616–636. URL `http://jmlr.csail.mit.edu/papers/v8/kalisch07a.html`. 512, 513

Kalisch, Markus, Martin Mächler and Diego Colombo (2010). *pcalg: Estimation of CPDAG/PAG and causal inference using the IDA algorithm*. URL `http://CRAN.R-project.org/package=pcalg`. R package version 1.1-2. 506, 606

Kalisch, Markus, Martin Mächler, Diego Colombo, Marloes H. Maathuis and Peter Bühlmnann (2012). "Causal Inference Using Graphical Models with the R Package pcalg." *Journal of Statistical Software*, **47(11)**: 1–26. URL `http://www.jstatsoft.org/v47/i11`. 506, 514, 606

Kallenberg, Wilbert C. M. and Teresa Ledwina (1997). "Data-Driven Smooth Tests When the Hypothesis Is Composite." *Journal of the American Statistical Association*, **92**: 1094–1104. URL `http://doc.utwente.nl/62408/`. 687, 704

Kalman, R. E. and R. S. Bucy (1961). "New Results in Linear Filtering and Prediction." *ASME Transactions, Journal of Basic Engineering*, **83D**: 95–108. 544

Kalman, Rudolf E. (1960). "A New Approach to Linear Filtering and Prediction Problems." *ASME Transactions, Journal of Basic Engineering*, **82D**: 35–50. 544

Kanai, Ryota, Tom Feilden, Colin Firth and Geraint Rees (2011). "Political Orientations Are Correlated with Brain Structure in Young Adults." *Current Biology*, **21**: 677–680. doi:10.1016/j.cub.2011.03.017. 697, 67

Kantz, Holger and Thomas Schreiber (2004). *Nonlinear Time Series Analysis*. Cambridge, England: Cambridge University Press, 2nd edn. 534, 550

Kao, Yi-hao and Benjamin Van Roy (2013). "Learning a Factor Model via Regularized PCA." *Machine Learning*, **91**: 279–303. URL `http://arxiv.org/abs/1111.6201`. doi:10.1007/s10994-013-5345-8. 385

Kaplan, E. L. and Paul Meier (1958). "Nonparametric Estimation from Incomplete Observations." *Journal of the American Statistical Association*, **53**: 457–481. URL `https://www.jstor.org/stable/2281868`. doi:10.2307/2281868. 742

Kass, Robert E. and Paul W. Vos (1997). *Geometrical Foundations of Asymptotic Inference*. New York: Wiley. 712

Kearns, Michael J. and Umesh V. Vazirani (1994). *An Introduction to Computational Learning Theory*. Cambridge, Massachusetts: MIT Press. 93

Kelly, Kevin T. (2007). "Ockham's razor, empirical complexity, and truth-finding efficiency." *Theoretical Computer Science*, **383**: 270–289. doi:10.1016/j.tcs.2007.04.009. 536

Kendall, Bruce E., Stephen P. Ellner, Edward Mccauley, Simon N. Wood, Cheryl J. Briggs, William W. Murdoch and Peter Turchin (2005). "Population Cycles in the Pine Looper Moth: Dynamical Tests of Mechanistic Hypotheses." *Ecological Monographs*, **75**: 259–276. URL https://escholarship.org/uc/item/2tq9h5tq. doi:10.1890/03-4056. 582

Khinchin, Aleksandr Iakovlevich (1957). *Mathematical Foundations of Information Theory*. New York: Dover. Translated by R. A. Silverman and M. D. Friedman from two Russian articles in Uspekhi Matematicheskikh Nauk, **7** (1953): 3–20 and **9** (1956): 17–75. 663

Kindermann, Ross and J. Laurie Snell (1980). *Markov Random Fields and their Applications*. Providence, Rhode Island: American Mathematical Society. URL http://www.ams.org/online_bks/conm1/. 450

King, Gary and Richard Nielsen (2016). "Why Propensity Scores Should Not Be Used for Matching." Electronic preprint. URL http://j.mp/1FQhySn. 496

King, Gary and Margaret E. Roberts (2015). "How Robust Standard Errors Expose Methodological Problems They Do Not Fix, and What to Do About It." *Political Analysis*, **23**: 159–179. URL http://gking.harvard.edu/publications/how-robust-standard-errors-expose-methodological-problems-they-do-not-fix. doi:10.1093/pan/mpu015. 167

Klein, Judy L. (1997). *Statistical Visions in Time: A History of Time Series Analysis, 1662–1938*. Cambridge, England: Cambridge University Press. 66, 543, 548

Knight, Frank B. (1975). "A Predictive View of Continuous Time Processes." *Annals of Probability*, **3**: 573–596. URL http://projecteuclid.org/euclid.aop/1176996302. 550

Koenker, Roger and Kevin F. Hallock (2001). "Quantile Regression." *Journal of Economic Perspectives*, **15**: 143–156. URL http://www.econ.uiuc.edu/~roger/research/rq/QRJEP.pdf. doi:10.1257/jep.15.4.143. 646

Kogan, Barry S. (1985). *Averroes and the Metaphysics of Causation*. Albany, New York: State University of New York Press. 456

Kolmogorov, Andrei N. (1941). "Interpolation und Extrapolation von stationären zufälligen Folgen." *Bulletin of the Academy Sciences, USSR, Mathematical Series*, **5**: 3–14. In Russian with German translation. 67

Koyama, Shinsuke, Lucia Castellanos Pérez-Bolde, Cosma Rohilla Shalizi and Robert E. Kass (2010). "Approximate Methods for State-Space Models." *Journal of the American Statistical Association*, **105**: 170–180. URL http://arxiv.org/abs/1004.3476. doi:10.1198/jasa.2009.tm08326. 548

Kpotufe, Samory (2011). "k-NN Regression Adapts to Local Intrinsic Dimension." In *Advances in Neural Information Processing Systems 24 [NIPS 2011]* (John Shawe-Taylor and Richard S. Zemel and Peter L. Bartlett and Fernando Pereira and Kilian Q. Weinberger, eds.), pp. 729–737. Cambridge, Massachusetts: MIT Press. URL http://papers.nips.cc/paper/4455-k-nn-regression-adapts-to-local-intrinsic-dimension. 489

Krige, D. G. (1981). *Lognormal-de Wijsian Geostatistics for Ore Evaluation*. South African Institute of Mining and Metallurgy Monograph Series. Johannesburg: South African Institute of Mining and Metallurgy, 2nd edn. URL http://www.saimm.co.za/Conferences/Geostatistics/GeostatisticsI.pdf. 67

Kullback, Solomon (1968). *Information Theory and Statistics*. New York: Dover Books, 2nd edn. 443, 676

Kullback, Solomon and R. A. Leibler (1951). "On Information and Sufficiency." *Annals of Mathematical Statistics*, **22**: 79–86. doi:10.1214/aoms/1177729694. 676

Künsch, Hans R. (1989). "The Jackknife and the Bootstrap for General Stationary Observations." *Annals of Statistics*, **17**: 1217–1241. URL http://projecteuclid.org/euclid.aos/1176347265. doi:10.1214/aos/1176347265. 532, 549

Lacerda, Gustavo, Peter Spirtes, Joseph Ramsey and Patrik Hoyer (2008). "Discovering Cyclic Causal Models by Independent Components Analysis." In *Proceedings of the Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence*

(UAI-08), pp. 366–374. Corvallis, Oregon: AUAI Press. URL http://uai.sis.pitt.edu/papers/08/p366-lacerda.pdf. 448, 514

Lahiri, S. N. (2003). *Resampling Methods for Dependent Data*. New York: Springer-Verlag. 532, 549

Lambert, Diane and Kathryn Roeder (1995). "Overdispersion Diagnostics for Generalized Linear Models." *Journal of the American Statistical Association*, **90**: 1225–1236. URL http://www.jstor.org/stable/2291513. 296

Landauer, Thomas K. and Susan T. Dumais (1997). "A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge." *Psychological Review*, **104**: 211–240. URL http://lsa.colorado.edu/papers/plato/plato.annote.html. 362, 368

Lange, Kenneth (2013). *Optimization*. New York: Springer, 2nd edn. 660

Langford, John, Ruslan Salakhutdinov and Tong Zhang (2009). "Learning Nonlinear Dynamic Models." In *Proceedings of the 26th Annual International Conference on Machine Learning [ICML 2009]* (Andrea Danyluk and Léon Bottou and Michael Littman, eds.), pp. 593–600. New York: Association for Computing Machinery. URL http://arxiv.org/abs/0905.3369. 550

Lasota, Andrzej and Michael C. Mackey (1994). *Chaos, Fractals, and Noise: Stochastic Aspects of Dynamics*. Berlin: Springer-Verlag. doi:10.1007/978-1-4612-4286-4. First edition, *Probabilistic Properties of Deterministic Systems*, Cambridge University Press, 1985. 480

Lauritzen, Steffen L. (1984). "Extreme Point Models in Statistics." *Scandinavian Journal of Statistics*, **11**: 65–91. URL http://www.jstor.org/pss/4615945. With discussion and response. 406

— (1996). *Graphical Models*. New York: Oxford University Press. 437, 448, 450, 460

Lawrie, Ian D. (1990). *A Unified Grand Tour of Theoretical Physics*. Bristol, England: Adam Hilger. 712

Lee, Ann B. and Larry Wasserman (2010). "Spectral Connectivity Analysis." *Journal of the American Statistical Association*, **105**: 1241–1255. URL http://arxiv.org/abs/0811.0121. doi:10.1198/jasa.2010.tm09754. 728

Lehmann, Erich L. (2008). "On the history and use of some standard statistical models." In *Probability and Statistics; Essays in Honor of David A. Freedman* (Deborah Nolan and Terry Speed, eds.), pp. 114–126. Brentwood, Ohio: Institute of Mathematical Statistics. URL http://projecteuclid.org/euclid.imsc/1207580081. doi:10.1214/193940307000000419. 66

Leisch, Friedrich (2004). "FlexMix: A General Framework for Finite Mixture Models and Latent Class Regression in R." *Journal of Statistical Software*, **11**. URL http://www.jstatsoft.org/v11/i08. 412

Li, Ching Chun (1975). *Path Analysis: A Primer*. Pacific Grove, California: The Boxwood Press. 450

Li, Ching Chun, Sati Mazumdar and B. Raja Rao (1975). "Partial Correlation in Terms of Path Coefficients." *The American Statistician*, **29**: 89–90. URL http://www.jstor.org/stable/2683271. 440

Li, Ming and Paul M. B. Vitányi (1997). *An Introduction to Kolmogorov Complexity and Its Applications*. New York: Springer-Verlag, 2nd edn. 467

Li, Qi and Jeffrey Scott Racine (2007). *Nonparametric Econometrics: Theory and Practice*. Princeton, New Jersey: Princeton University Press. 124, 210, 224, 494, 500

Ligges, Uwe and Martin Mächler (2003). "scatterplot3d: An R Package for Visualizing Multivariate Data." *Journal of Statistical Software*, **8(11)**: 1–20. URL http://www.jstatsoft.org/v8/i11/. 606

Lindsey, J. K. (2004). *Statistical Analysis of Stochastic Processes in Time*. Cambridge, England: Cambridge University Press. 549

Little, Roderick J. A. and Donald B. Rubin (1987). *Statistical Analysis with Missing Data*. New York: John Wiley and Sons. 755

Littman, Michael L., Richard S. Sutton and Satinder Singh (2002). "Predictive Representations of State." In *Advances in Neural Information Processing Systems 14 (NIPS 2001)* (Thomas G. Dietterich and Suzanna Becker and Zoubin Ghahramani, eds.), pp. 1555–1561. Cambridge, Massachusetts: MIT Press. URL `http://papers.nips.cc/paper/1983-predictive-representations-of-state`. 550

Liu, Han, John Lafferty and Larry Wasserman (2009). "The Nonparanormal: Semiparametric Estimation of High Dimensional Undirected Graphs." *Journal of Machine Learning Research*, **10**: 2295–2328. URL `http://jmlr.csail.mit.edu/papers/v10/liu09a.html`. 448

Liu, Ka-Yuet, Marissa King and Peter S. Bearman (2010). "Social Influence and the Autism Epidemic." *American Journal of Sociology*, **115**: 1387–1434. URL `http://www.understandingautism.columbia.edu/papers/social-influence-and-the-autism-epidemic-(2010).pdf`. doi:10.1086/651448. 495

Loehlin, John C. (1992). *Latent Variable Models: An Introduction to Factor, Path, and Structural Analysis*. Hillsdale, New Jersey: Lawrence Erlbaum Associates, 2nd edn. 401, 450

Loève, Michel (1955). *Probability Theory*. New York: D. Van Nostrand Company, 1st edn. 368

Lunde, Robert (2018). *Bootstrapping and Sample Splitting Under Weak Dependence*. Ph.D. thesis, Carnegie Mellon University. 93

Lunde, Robert and Cosma Rohilla Shalizi (2017). "Bootstrapping Generalization Error Bounds for Time Series." arxiv:1711.02834. URL `https://arxiv.org/abs/1711.02834`. 549

Maathuis, Marloes H., Diego Colombo, Markus Kalisch and Peter Bühlmann (2010). "Predicting Causal Effects in Large-scale Systems from Observational Data." *Nature Methods*, **7**: 247–248. URL `http://stat.ethz.ch/Manuscripts/buhlmann/maathuisetal2010.pdf`. doi:10.1038/nmeth0410-247. See also `http://stat.ethz.ch/Manuscripts/buhlmann/maathuisetal2010SI.pdf`. 514

Maathuis, Marloes H., Markus Kalisch and Peter Bühlmann (2009). "Estimating High-Dimensional Intervention Effects from Observational Data." *Annals of Statistics*, **37**: 3133–3164. URL `http://arxiv.org/abs/0810.4214`. doi:10.1214/09-AOS685. 513

MacCulloch, Diarmaid (2004). *The Reformation: A History*. New York: Penguin. 91

MacKenzie, Donald (2006). *An Engine, Not a Camera: How Financial Models Shape Markets*. Cambridge, Massachusetts: MIT Press. 30

Maguire, B. A., E. S. Pearson and A. H. A. Wynn (1952). "The Time Intervals between Industrial Accidents." *Biometrika*, **39**: 168–180. URL `http://www.jstor.org/pss/2332475`. 420

Mahoney, Michael W. (2011). "Randomized Algorithms for Matrices and Data." *Foundations and Trends in Machine Learning*, **2**: 123–224. URL `https://arxiv.org/abs/1104.5557`. doi:10.1561/2200000035. 368

Mandelbrot, Benoit (1962). "The Role of Sufficiency and of Estimation in Thermodynamics." *Annals of Mathematical Statistics*, **33**: 1021–1038. URL `http://projecteuclid.org/euclid.aoms/1177704470`. 295, 450

Manski, Charles F. (2003). *Partial Identification of Probability Distributions*. New York: Springer-Verlag. 755

— (2007). *Identification for Prediction and Decision*. Cambridge, Massachusetts: Harvard University Press. 465, 482, 513, 745, 755

Marcellesi, Alexandre (2013). "Is Race a Cause?" *Philosophy of Science*, **80**: 650–659. doi:10.1086/673721. 487

Martindale, Colin (1990). *The Clockwork Muse: The Predictability of Artistic Change*. New York: Basic Books. 544

Matheron, Georges (2019). *Matheron's Theory of Regionalised Variables*. Oxford: Oxford University Press. doi:10.1093/oso/9780198835660.001.0001. Based on lecture notes from 1970, translated from French into English by Charles Huijbregts. 67

Matloff, Norman (2011). *The Art of R Programming: A Tour of Statistical Software Design*. San Francisco: No Starch Press. 796

McGee, Leonard A. and Stanley F. Schmidt (1985). *Discovery of the Kalman Filter as a Practical Tool for Aerospace and Industry*. Tech. Rep. 86847, NASA Technical Memorandum. URL `https://ntrs.nasa.gov/citations/19860003843`. 422, 549

McLachlan, Geoffrey J. and Thriyambakam Krishnan (2008). *The EM Algorithm and Extensions*. Hoboken, New Jersey: John Wiley and Sons. 755

Metropolis, Nicholas, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller and Edward Teller (1953). "Equations of State Calculations by Fast Computing Machines." *Journal of Chemical Physics*, **21**: 1087–1092. doi:10.1063/1.1699114. 131

Miller, John H. (1998). "Active Nonlinear Tests (ANTs) of Complex Simulation Models." *Management Science*, **44**: 820–830. URL `http://zia.hss.cmu.edu/miller/papers/antabst.html`. 138

Mitchell, Tom M. (1997). *Machine Learning*. New York: McGraw-Hill. 315

Mohan, Karthika, Judea Pearl and Jin Tian (2013). "Graphical Models for Inference with Missing Data." In *Advances in Neural Information Processing Systems 26 [NIPS 2013]* (C. J. C. Burges and Léon Bottou and Max Welling and Zoubin Ghahramani and Kilian Q. Weinberger, eds.), pp. 1277–1285. Red Hook, New York: Curran Associates. URL `http://papers.nips.cc/paper/4899-graphical-models-for-inference-with-missing-data`. 748

Mohri, Mehryar, Afshin Rostamizadeh and Ameet Talwalkar (2012). *Foundations of Machine Learning*. Cambridge, Massachusetts: MIT Press. 93

Monahan, John F. (2001). *Numerical Methods of Statistics*. Cambridge, England: Cambridge University Press. 137

Moran, P. A. P. (1961). "Path Coefficients Reconsidered." *Australian Journal of Statistics*, **3**: 87–93. doi:10.1111/j.1467-842X.1961.tb00314.x. 450

Morgan, Stephen L. and Christopher Winship (2007). *Counterfactuals and Causal Inference: Methods and Principles for Social Research*. Cambridge, England: Cambridge University Press. 460, 461, 472, 483, 495

— (2015). *Counterfactuals and Causal Inference: Methods and Principles for Social Research*. Cambridge, England: Cambridge University Press, 2nd edn. 460, 473

Nadaraya, E. A. (1964). "On Estimating Regression." *Theory of Probability and Its Applications*, **9**: 141–142. doi:10.1137/1109020. 43, 124

Neal, Radford M. and Geoffrey E. Hinton (1998). "A View of the EM Algorithm that Justifies Incremental, Sparse, and Other Variants." In Jordan (1998), pp. 355–368. URL `http://www.cs.toronto.edu/~radford/em.abstract.html`. 422

Newey, Whitney K. and James L. Powell (2003). "Instrumental Variable Estimation of Nonparametric Models." *Econometrica*, **71**: 1565–1578. doi:10.1111/1468-0262.00459. 483

Newman, Mark E. J. and G. T. Barkema (1999). *Monte Carlo Methods in Statistical Physics*. Oxford: Clarendon Press. 138

Novembre, John and Matthew Stephens (2008). "Interpreting principal component analyses of spatial population genetic variation." *Nature Genetics*, **40**: 646–649. doi:10.1038/ng.139. 366, 370, 371

Packard, Norman H., James P. Crutchfield, J. Doyne Farmer and Robert S. Shaw (1980). "Geometry from a Time Series." *Physical Review Letters*, **45**: 712–716. doi:10.1103/PhysRevLett.45.712. 549

Paige, Robert L. and A. Alexandre Trindade (2010). "The Hodrick-Prescott Filter: A special case of penalized spline smoothing." *Electronic Journal of Statistics*, **4**: 856–874. URL `http://projecteuclid.org/euclid.ejs/1284557751`. doi:10.1214/10-EJS570. 183

Palomar, Daniel P. and Sergio Verdú (2008). "Lautum Information." *IEEE Transactions on Information Theory*, **54**: 964–975. URL `http://www.princeton.edu/~verdu/lautum.info.pdf`. 443, 670

Parzen, Emanuel (1962). "On Estimation of a Probability Density Function and Mode." *Annals of Mathematical Statistics*, **33**: 1065–1076. URL `https://projecteuclid.org/euclid.aoms/1177704472`. doi:10.1214/aoms/1177704472. 345

Pearl, Judea (1988). *Probabilistic Reasoning in Intelligent Systems*. New York: Morgan Kaufmann. 450

— (2000). *Causality: Models, Reasoning, and Inference*. Cambridge, England: Cambridge University Press. 437, 450

— (2009a). "Causal inference in statistics: An overview." *Statistics Surveys*, **3**: 96–146. URL `http://projecteuclid.org/euclid.ssu/1255440554`. doi:10.1214/09-SS057. 460, 461, 492

— (2009b). *Causality: Models, Reasoning, and Inference*. Cambridge, England: Cambridge University Press, 2nd edn. 450, 460, 469, 472, 473, 478, 480, 483, 492, 503

Pearson, Karl (1901). "On lines and planes of closest fit to systems of points in space." *Philosophical Magazine*, **2 (series 6)**: 559–572. doi:10.1080/14786440109462720. 368

Pelillo, Marcello (2014). "Alhazen and the nearest neighbor rule." *Pattern Recognition Letters*, **38**: 34–37. doi:https://doi.org/10.1016/j.patrec.2013.10.022. 43

Peterson, Robert A. (2000). "A Meta-Analysis of Variance Accounted for and Factor Loadings in Exploratory Factor Analysis." *Marketing Letters*, **11**: 261–275. 389

Pitman, E. J. G. (1979). *Some Basic Theory for Statistical Inference*. London: Chapman and Hall. 324

Pollard, David (1989). "Asymptotics via Empirical Processes." *Statistical Science*, **4**: 341–354. URL `http://projecteuclid.org/euclid.ss/1177012394`. doi:10.1214/ss/1177012394. 324, 344

— (1990). *Empirical Processes: Theory and Applications*, vol. 2 of *NSF-CBMS Regional Conference Series in Probability and Statistics*. Hayward, California: Institute of Mathematical Statistics. URL `http://www.stat.yale.edu/~pollard/`. 344

Porter, Theodore M. (1986). *The Rise of Statistical Thinking, 1820–1900*. Princeton, New Jersey: Princeton University Press. 19, 66

Press, William H., Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery (1992). *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge, England: Cambridge University Press, 2nd edn. URL `http://www.nrbook.com/`. 137

Puccia, Charles J. and Richard Levins (1985). *Qualitative Modeling of Complex Systems: An Introduction to Loop Analysis and Time Averaging*. Cambridge, Massachusetts: Harvard University Press. 448

Quiñonero-Candela, Joaquin, Masashi Sugiyama, Anton Schwaighofer and Neil D. Lawrence (eds.) (2009). *Dataset Shift in Machine Learning*. Cambridge, Massachusetts: MIT Press. 89, 226

R Core Team (2015a). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL `http://www.R-project.org`. ISBN 3-900051-07-0. 606

— (2015b). *`foreign`: Read Data Stored by Minitab, S, SAS, SPSS, Stata, Systat, Weka, dBase, ...*. URL `https://CRAN.R-project.org/package=foreign`. R package version 0.8-66. 606

Racine, Jeff (2000). "Consistent cross-validatory model-selection for dependent data: *hv*-block cross-validation." *Journal of Econometrics*, **99**: 39–61. doi:10.1016/S0304-4076(00)00030-0. 549

Racine, Jeffrey S. (2008). "Nonparametric Econometrics: A Primer." *Foundations and Trends in Econometrics*, **3**: 1–88. URL `http://socserv.mcmaster.ca/racine/ECO0301.pdf`. doi:10.1561/0800000009. 124

Raginsky, Maxim (2011). "Directed Information and Pearl's Causal Calculus." In *Proceedings of the 49th Annual Allerton Conference on Communication, Control and Computing* (S. Meyn and B. Hajek, eds.), pp. 958–965. IEEE. URL `http://arxiv.org/abs/1110.0718`. doi:10.1109/Allerton.2011.6120270. 461

Rayner, J. C. W. and D. J. Best (1989). *Smooth Tests of Goodness of Fit*. Oxford: Oxford University Press. 684, 704

Reichenbach, Hans (1956). *The Direction of Time*. Berkeley: University of California Press. Edited by Maria Reichenbach. 501, 503

Reid, Constance (1982). *Neyman from Life*. New York: Springer-Verlag. doi:10.1007/978-1-4612-5754-7. 315

Reinhart, Abiel (2014). *pdfetch: Fetch economic and financial time series data from public sources*. URL https://CRAN.R-project.org/package=pdfetch. R package version 0.1.6. 606

Reinsch, Christian H. (1967). "Smoothing by Spline Functions." *Numerische Mathematik*, **10**: 177–183. 183

Rényi, Alfréd (1961). "On Measures of Entropy and Information." In *Proceedings of the Fourth Berkeley Symposium on Mathematical Statistics and Probability* (Jerzy Neyman, ed.), vol. 1, pp. 547–561. Berkeley: University of California Press. URL https://projecteuclid.org/proceedings/berkeley-symposium-on-mathematical-statistics-and-probability/Proceedings-of-the-Fourth-Berkeley-Symposium-on-Mathematical-Statistics-and/Chapter/On-Measures-of-Entropy-and-Information/bsmsp/1200512181. 663

Richardson, Thomas (1996). "A Discovery Algorithm for Directed Cyclic Graphs." In *Proceedings of the Proceedings of the Twelfth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-96)*, pp. 454–446. San Francisco, CA: Morgan Kaufmann. URL ftp://ftp.andrew.cmu.edu/pub/phil/thomas/TR68.ps. URL is for expanded version. 448, 514

Richardson, Thomas S. and James M. Robins (2013). *Single World Intervention Graphs (SWIGs): A Unification of the Counterfactual and Graphical Approaches to Causality*. Tech. Rep. 128, Center for Statistics and the Social Sciences, University of Washington. URL http://www.csss.washington.edu/Papers/wp128.pdf. 461

Ripley, Brian D. (1996). *Pattern Recognition and Neural Networks*. Cambridge, England: Cambridge University Press. 315, 317

— (2015). *tree: Classification and Regression Trees*. URL https://CRAN.R-project.org/package=tree. R package version 1.0-36. 300, 606

Robert, Christian P. and George Casella (2004). *Monte Carlo Statistical Methods*. New York: Springer, 2nd edn. 138

Robins, James M., Richard Scheines, Peter Spirtes and Larry Wasserman (2003). "Uniform Consistency in Causal Inference." *Biometrika*, **90**: 491–515. URL http://www.stat.cmu.edu/tr/tr725/tr725.html. 512

Rockafellar, R. T. (1970). *Convex Analysis*. Princeton: Princeton University Press. 677

Rodrik, Dani (2008). "The Real Exchange Rate and Economic Growth." *Brookings Papers on Economic Activity*, **2008(2)**: 365–412. URL http://www.hks.harvard.edu/fs/drodrik/Research%20papers/RER%20and%20growth.pdf. doi:10.1353/eca.0.0020. 9

Rosenbaum, Paul and Donald Rubin (1983). "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika*, **70**: 41–55. URL http://www.jstor.org/stable/2335942. 491, 496

Rosenblatt, Murray (1956). "Remarks on Some Nonparametric Estimates of a Density Function." *Annals of Mathematical Statistics*, **27**: 832–837. URL https://projecteuclid.org/euclid.aoms/1177728190. doi:10.1214/aoms/1177728190. 345

Rosenzweig, Mark R. and Kenneth I. Wolpin (2000). "Natural "Natural Experiments" in Economics." *Journal of Economic Literature*, **38**: 827–874. doi:10.1257/jel.38.4.827. 478, 483

Roweis, Sam T. and Laurence K. Saul (2000). "Nonlinear Dimensionality Reduction by Locally Linear Embedding." *Science*, **290**: 2323–2326. doi:10.1126/science.290.5500.2323. 713

Rubin, Donald B. (1987). *Multiple Imputation for Nonresponse in Surveys*. New York: Wiley. 750, 755

— (2006). *Matched Sampling for Causal Effects*. Cambridge, England: Cambridge University Press. 461, 496

Rubin, Donald B. and Richard P. Waterman (2006). "Estimating the Causal Effects of Marketing Interventions Using Propensity Score Methodology." *Statistical Science*, **21**: 206–222. URL http://arxiv.org/abs/math.ST/0609201. doi:10.1214/088342306000000259. 496

Ruelle, David (1991). *Chance and Chaos*. Princeton, New Jersey: Princeton University Press. doi:10.2307/j.ctv10crf7w. 550

Russell, Bertrand (1920). *Introduction to Mathematical Philosophy*. London: George Allen and Unwin, 2nd edn. URL http://people.umass.edu/klement/russell-imp.html. First edition, 1919. 137

— (1927). *The Analysis of Matter*. London: K. Paul Trench, Trubner and Co. Reprinted New York: Dover Books, 1954. 503

Salmon, Wesley C. (1984). *Scientific Explanation and the Causal Structure of the World*. Princeton: Princeton University Press. doi:10.1515/9780691221489. 483, 484

Sandhaus, Evan (2008). "The New York Times Annotated Corpus." Electronic database. URL http://www.ldc.upenn.edu/Catalog/CatalogEntry.jsp?catalogId=LDC2008T19. 362

Sarkar, Deepayan (2008). *Lattice: Multivariate Data Visualization with R*. New York: Springer. URL http://lmdvr.r-forge.r-project.org. 606

Saul, Lawrence K. and Sam T. Roweis (2003). "Think Globally, Fit Locally: Supervised Learning of Low Dimensional Manifolds." *Journal of Machine Learning Research*, **4**: 119–155. URL http://jmlr.csail.mit.edu/papers/v4/saul03a.html. 713

Schoenberg, I. J. (1964). "Spline Functions and the Problem of Graduation." *Proceedings of the National Academy of Sciences (USA)*, **52**: 947–950. 183

Schreiber, Thomas (1997). "Detecting and Analysing Nonstationarity in a Time Series with Nonlinear Cross Prediction." *Physics Letters A*, **160**: 411. URL http://arxiv.org/abs/chao-dyn/9909044. 534

Schutz, Bernard F. (1980). *Geometrical Methods of Mathematical Physics*. Cambridge, England: Cambridge University Press. 712

Schwartz, David N. (2017). *The Last Man Who Knew Everything: The Life and Times of Enrico Fermi, Father of the Nuclear Age*. New York: Basic Books. 132

Schwarz, Gideon (1978). "Estimating the Dimension of a Model." *Annals of Statistics*, **6**: 461–464. URL http://projecteuclid.org/euclid.aos/1176344136. 687

Scott, Clayton and Robert Nowak (2005). "A Neyman-Pearson Approach to Statistical Learning." *IEEE Transactions on Information Theory*, **51**: 3806–3819. URL http://www.ece.wisc.edu/~nowak/np.pdf. doi:10.1109/TIT.2005.856955. 314

Seidenfeld, Teddy (1987). "Entropy and Uncertainty." In *Foundations of Statistical Inference* (I. B. MacNeill and G. J. Umphrey, eds.), pp. 259–287. Dordrecht: D. Reidel. 664

Self, Steven G. and Kung-Yee Liang (1987). "Asymptotic Properties of Maximum Likelihood Estimators and Likelihood Ratio Tests Under Nonstandard Conditions." *Journal of the American Statistical Association*, **82**: 605–610. URL http://www.jstor.org/stable/2289471. doi:10.2307/2289471. 660

Sethna, James P. (2006). *Statistical Mechanics: Entropy, Order Parameters, and Complexity*. Oxford: Oxford University Press. URL http://pages.physics.cornell.edu/sethna/StatMech/. 450, 676

Shalizi, Cosma Rohilla (2003). "Optimal Nonlinear Prediction of Random Fields on Networks." *Discrete Mathematics and Theoretical Computer Science*, **AB(DMCS)**: 11–30. URL http://arxiv.org/abs/math.PR/0305160. doi:10.46298/dmtcs.2310. 550

— (2007). "Maximum Likelihood Estimation and Model Testing for q-Exponential Distributions." E-print, arxiv:math.ST/0701854. URL http://arxiv.org/abs/math.ST/0701854. 420

— (2016). "Review of Imbens and Rubin (2015)." *Journal of the American Statistical Association*, **111**: 1364–1365. doi:10.1080/01621459.2016.1235436. 461, 496

Shalizi, Cosma Rohilla and James P. Crutchfield (2001). "Computational Mechanics: Pattern and Prediction, Structure and Simplicity." *Journal of Statistical Physics*, **104**: 817–879. URL http://arxiv.org/abs/cond-mat/9907176. doi:10.1023/A:1010388907793. 550

Shalizi, Cosma Rohilla and Andrew C. Thomas (2011). "Homophily and Contagion Are Generically Confounded in Observational Social Network Studies." *Sociological Methods and Research*, **40**: 211–239. URL http://arxiv.org/abs/1004.4704. doi:10.1177/0049124111404820. 482

Shannon, Claude E. (1948). "A Mathematical Theory of Communication." *Bell System Technical Journal*, **27**: 379–423. Reprinted in Shannon and Weaver (1963). 443, 676, 677

Shannon, Claude E. and Warren Weaver (1963). *The Mathematical Theory of Communication*. Urbana, Illinois: University of Illinois Press. 601

Shields, Paul C. (1996). *The Ergodic Theory of Discrete Sample Paths*. Providence, Rhode Island: American Mathematical Society. 549

Shpitser, Ilya and Judea Pearl (2008). "Complete Identification Methods for the Causal Hierarchy." *Journal of Machine Learning Research*, **9**: 1941–1979. URL http://jmlr.csail.mit.edu/papers/v9/shpitser08a.html. 480

Shumway, Robert H. and David S. Stoffer (2000). *Time Series Analysis and Its Applications*. New York: Springer-Verlag. 546, 548, 549, 577

Silverman, B. W. (1984). "Spline Smoothing: The Equivalent Variable Kernel Method." *Annals of Statistics*, **12**: 898–916. doi:10.1214/aos/1176346710. 180

— (1985). "Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting." *Journal of the Royal Statistical Society B*, **47**: 1–52. URL http://www.jstor.org/stable/2345542. 183

Simonoff, Jeffrey S. (1996). *Smoothing Methods in Statistics*. Berlin: Springer-Verlag. 43, 105, 124, 180, 182, 256, 343

Singh, Rahul, Maneesh Sahani and Arthur Gretton (2019). "Kernel Instrumental Variable Regression." In *Advances in Neural Information Processing Systems 32 [NeurIPS 2019]* (H. Wallach and H. Larochelle and A. Beygelzimer and F. d'Alché Buc and E. Fox and R. Garnett, eds.), pp. 4595–4607. Red Hook, New York: Curran Associates. URL https://arxiv.org/abs/1906.00232. 473

Solow, Robert M. (1970). *Growth Theory: An Exposition*. Oxford: Oxford University Press. New edition with the 1987 Nobel lecture. 535

Spain, Seth M., Kristin L. Sotak, Joey (Chou-Yu) Tsai, P. D. Harms and Sean T. Hannah (2012). "Testing the Form of Theoretical Models by Relaxing Assumptions: Comparing Parametric and Nonparametric Models." Electronic preprint, SSRN/2164297. URL http://ssrn.com/abstract=2164297. 224

Spanos, Aris (2011). "A Frequentist Interpretation of Probability for Model-based Inductive Inference." *Synthese*, **190**. doi:10.1007/s11229-011-9892-x. 536

Spearman, Charles (1904). ""General Intelligence," Objectively Determined and Measured." *American Journal of Psychology*, **15**: 201–293. URL http://psychclassics.yorku.ca/Spearman/. 377, 401

Spector, Phil (2008). *Data Manipulation with R*. Berlin: Springer. 796

Spector, Phil, Jerome Friedman, Robert Tibshirani and Thomas Lumley (2013). *acepack: ace() and avas() for selecting regression transformations*. URL http://CRAN.R-project.org/package=acepack. R package version 1.3-3.3. 60

Spirtes, Peter, Clark Glymour and Richard Scheines (1993). *Causation, Prediction, and Search*. Berlin: Springer-Verlag, 1st edn. doi:10.1007/978-1-4612-2748-9. 450

— (2001). *Causation, Prediction, and Search*. Cambridge, Massachusetts: MIT Press, 2nd edn. 437, 450, 460, 461, 503, 505, 506, 510, 512, 513

Spivak, Michael (1965). *Calculus on Manifolds: A Modern Approach to Classical Theorems of Advanced Calculus*. Menlo Park, California: Benjamin Cummings. 712

Spufford, Francis (2010). *Red Plenty*. London: Faber and Faber. 660, 677

Sriperumbudur, Bharath K., Arthur Gretton, Kenji Fukumizu, Bernhard Schölkopf and Gert R.G. Lanckriet (2010). "Hilbert Space Embeddings and Metrics on Probability Measures." *Journal of Machine Learning Research*, **11**: 1517–1561. URL http://jmlr.csail.mit.edu/papers/v11/sriperumbudur10a.html. 499

Stephan, Maria J. and Erica Chenowth (2008). "Why Civil Resistance Works; The Strategic Logic of Nonviolent Conflict." *International Security*, **33**: 7–44. doi:10.1162/isec.2008.33.1.7. 25

Stigler, Stephen M. (1986). *The History of Statistics: The Measurement of Uncertainty before 1900*. Cambridge, Massachusetts: Harvard University Press. 19, 66

Stone, M. (1974). "Cross-validatory choice and assessment of statistical predictions." *Journal of the Royal Statistical Society B*, **36**: 111–147. URL `http://www.jstor.org/stable/2984809`. 93

Stuart, Elizabeth A. (2010). "Matching Methods for Causal Inference: A Review and a Look Forward." *Statistical Science*, **25**: 1–21. URL `http://arxiv.org/abs/1010.5586`. doi:10.1214/09-STS313. 495, 496

Sung, Yun Ju and Charles J. Geyer (2007). "Monte Carlo Likelihood Inference for Missing Data Models." *Annals of Statistics*, **35**: 990–1011. doi:10.1214/009053606000001389. 755

Székely, Gábor J. and Maria L. Rizzo (2009). "Brownian Distance Covariance." *Annals of Applied Statistics*, **3**: 1236–1265. URL `http://arxiv.org/abs/1010.0297`. doi:10.1214/09-AOAS312. With discussion and reply. 499

Taylor, G. I. (1922). "Diffusion by Continuous Movements." *Proceedings of the London Mathematical Society*, **20**: 196–212. doi:10.1112/plms/s2-20.1.196. 549

Taylor, Jonathan and Robert J. Tibshirani (2015). "Statistical Learning and Selective Inference." *Proceedings of the National Academy of Sciences (USA)*, **112**: 7629–7634. doi:10.1073/pnas.1507583112. 93

Therneau, Terry M. (2015). *A Package for Survival Analysis in S*. URL `https://CRAN.R-project.org/package=survival`. Version 2.38. 742

Thiesson, Bo, David Maxwell Chickering, David Heckerman and Christopher Meek (2004). "ARMA Time-Series Modeling with Graphical Models." In *Uncertainty in Artificial Intelligence: Proceedings of the Twentieth Conference (UAI 2004)* (Max Chickering and Joseph Y. Halpern, eds.), pp. 552–560. Arlington, Virginia: AUAI Press. URL `http://arxiv.org/abs/1207.4162`. 546

Thomson, Godfrey H. (1916). "A Hierarchy without a General Factor." *British Journal of Psychology*, **8**: 271–281. doi:10.1111/j.2044-8295.1916.tb00133.x. 397

— (1939). *The Factorial Analysis of Human Ability*. Boston: Houghton Mifflin Company. URL `http://www.archive.org/details/factorialanalysi032965mbp`. 368, 401

Thurstone, L. L. (1934). "The Vectors of Mind." *Psychological Review*, **41**: 1–32. URL `http://psychclassics.yorku.ca/Thurstone/`. 401

Tibshirani, Robert (1996). "Regression Shrinkage and Selection via the Lasso." *Journal of the Royal Statistical Society B*, **58**: 267–288. URL `https://www.jstor.org/stable/2346178`. doi:10.1111/j.2517-6161.1996.tb02080.x. 643

Tibshirani, Ryan J., Alessandro Rinaldo, Robert Tibshirani and Larry Wasserman (2015). *Uniform Asymptotic Inference and the Bootstrap After Model Selection*. Tech. rep., Statistics Department, Carnegie Mellon University. URL `http://arxiv.org/abs/1506.06266`. 93

Tibshirani, Ryan J. and Robert Tibshirani (2009). "A Bias Correction for the Minimum Error Rate in Cross-Validation." *Annals of Applied Statistics*, **3**: 822–829. URL `http://arxiv.org/abs/0908.2904`. 85, 103

Tilly, Charles (1984). *Big Structures, Large Processes, Huge Comparisons*. New York: Russell Sage Foundation. 483

— (1998). *Durable Inequality*. Berkeley: University of California Press. 482

— (2008). *Explaining Social Processes*. Boulder, Colorado: Paradigm Publishers. 483

Trapletti, Adrian and Kurt Hornik (2015). *tseries: Time Series Analysis and Computational Finance*. URL `http://CRAN.R-project.org/package=tseries`. R package version 0.10-34. 606

Tsai, Yao-Hung Hubert, Shaojie Bai, Makoto Yamada, Louis-Philippe Morency and Ruslan Salakhutdinov (2019). "Transformer Dissection: A Unified Understanding of Transformer's Attention via the Lens of Kernel." In *Proceedings of the 2019 Conference on Empirical*

Methods in Natural Language Processing [EMNLP 2019], pp. 4344–4353. URL http://arxiv.org/abs/1908.11775. doi:10.18653/v1/D19-1443. 124

Tukey, John W. (1954). "Unsolved Problems of Experimental Statistics." *Journal of the American Statistical Association*, **49**: 706–731. URL http://www.jstor.org/pss/2281535. 374

Tutz, Gerhard (2012). *Regression for Categorical Data*. Cambridge, England: Cambridge University Press. 296

van de Geer, Sara A. (2000). *Empirical Processes in M-Estimation*. Cambridge, England: Cambridge University Press. 344

van der Vaart, A. W. (1998). *Asymptotic Statistics*. Cambridge, England: Cambridge University Press. doi:10.1017/CBO9780511802256. 165, 660, 671

Vapnik, Vladimir N. (2000). *The Nature of Statistical Learning Theory*. Berlin: Springer-Verlag, 2nd edn. 93

Varadhan, Ravi (2012). *alabama: Constrained nonlinear optimization*. URL http://CRAN.R-project.org/package=alabama. R package version 2011.9-1. 645

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser and Illia Polosukhin (2017). "Attention Is All You Need." In *Advances in Neural Information Processing Systems 30 [NIPS 2017]* (I. Guyon and U. Von Luxburg and S. Bengio and H. Wallach and R. Fergus and S. Vishwanathan and R. Garnett, eds.), pp. 5998–6008. Red Hook, New York: Curran Associates. URL http://arxiv.org/abs/1706.03762. 124

Venables, W. N. and B. D. Ripley (2002). *Modern Applied Statistics with S*. Berlin: Springer-Verlag, 4th edn. URL http://www.stats.ox.ac.uk/pub/MASS4. 287, 606

Vidyasagar, Mathukumalli (2003). *Learning and Generalization: With Applications to Neural Networks*. Berlin: Springer-Verlag, 2nd edn. 93

von Luxburg, Ulrike and Bernhard Schölkopf (2008). "Statistical Learning Theory: Models, Concepts, and Results." arxiv:0810.4752. URL http://arxiv.org/abs/0810.4752. 93

Wahba, Grace (1990). *Spline Models for Observational Data*. Philadelphia: Society for Industrial and Applied Mathematics. 43, 182, 207

Wang, Lawrence (2016). *Network Comparisons using Sample Splitting*. Ph.D. thesis, Carnegie Mellon University. 93

Ward, Michael D., Brian D. Greenhill and Kristin M. Bakke (2010). "The Perils of Policy by p-value: Predicting Civil Conflicts." *Journal of Peace Research*, **47**: 363–375. doi:10.1177/0022343309356491. 22

Wasserman, Larry (2003). *All of Statistics: A Concise Course in Statistical Inference*. Berlin: Springer-Verlag. 43

— (2006). *All of Nonparametric Statistics*. Berlin: Springer-Verlag. 43, 105, 124, 182, 189, 256, 327, 343, 344

Watson, Geoffrey S. (1964). "Smooth Regression Analysis." *Sanhkya*, **26**: 359–372. URL http://www.jstor.org/stable/25049340. 43, 124

Weisberg, Sanford (1985). *Applied Linear Regression*. New York: Wiley, 2nd edn. 66

Welch, Lloyd R. (December 2003). "Shannon Lecture: Hidden Markov Models and the Baum-Welch Algorithm." *IEEE Information Theory Society Newsletter*, **53(4)**: 1 and 10–13. URL http://www-bcf.usc.edu/~lototsky/MATH508/Baum-Welch.pdf. 548

Western, Bruce (1996). "Vague Theory and Model Uncertainty in Macrosociology." *Sociological Methodology*, **26**: 165–192. URL http://www.jstor.org/stable/271022. doi:10.2307/271022. 84

White, Halbert (1994). *Estimation, Inference and Specification Analysis*. Cambridge, England: Cambridge University Press. doi:10.1017/CCOL0521252806. 93, 224, 652, 653, 656, 657, 660

Whittaker, E. T. (1922). "On a New Method of Graduation." *Proceedings of the Edinburgh Mathematical Society*, **41**: 63–75. doi:10.1017/S001309150000359X. 183

Wickham, Hadley (2011). "The Split-Apply-Combine Strategy for Data Analysis." *Journal of Statistical Software*, **40(1)**: 1–29. URL http://www.jstatsoft.org/v40/i01/. 606

— (2015). *Advanced R*. Boca Raton, Florida: CRC Press. 796

Wiener, Norbert (1949). *Extrapolation, Interpolation, and Smoothing of Stationary Time Series: With Engineering Applications*. Cambridge, Massachusetts: The Technology Press of the Massachusetts Institute of Technology. doi:10.7551/mitpress/2946.001.0001. "First published during the war [1942] as a classifed report to Section D_2, National Defense Research Council". 67, 546

— (1956). "Nonlinear Prediction and Dynamics." In *Proceedings of the Third Berkeley Symposium on Mathematical Statistics and Probability* (Jerzy Neyman, ed.), vol. 3, pp. 247–252. Berkeley: University of California Press. URL `http://projecteuclid.org/euclid.bsmsp/1200502197`. 331

— (1961). *Cybernetics: Or, Control and Communication in the Animal and the Machine*. Cambridge, Massachusetts: MIT Press, 2nd edn. First edition New York: Wiley, 1948. 503

Winkler, Gerhard (1995). *Image Analysis, Random Fields and Dynamic Monte Carlo Methods: A Mathematical Introduction*. Berlin: Springer-Verlag. 450

Wood, Simon N. (2004). "Stable and efficient multiple smoothing parameter estimation for generalized additive models." *Journal of the American Statistical Association*, **99**: 673–686. URL `http://www.maths.bath.ac.uk/~sw283/simon/papers/magic.pdf`. 606

— (2006). *Generalized Additive Models: An Introduction with R*. Boca Raton, Florida: Chapman and Hall/CRC. 182, 206, 296

— (2010). "Statistical inference for noisy nonlinear ecological dynamic systems." *Nature*, **466**: 1102–1104. doi:10.1038/nature09319. 582

Wright, Sewall (1934). "The Method of Path Coefficients." *Annals of Mathematical Statistics*, **5**: 161–215. URL `http://projecteuclid.org/euclid.aoms/1177732676`. 450

Wysocki, W. (1992). "Mathematical Foundations of Multivariate Path Analysis." *Inventiones Mathematicae*, **21**: 387–397. URL `https://eudml.org/doc/263277`. 450

Xie, Yihui (2015). *Dynamic Documents with R and knitr*. Boca Raton, Florida: CRC Press, 2nd edn. URL `http://yihui.name/knitr/`. 606

Yates, JoAnne (1989). *Control through Communication: The Rise of System in American Management*. Baltimore: Johns Hopkins University Press. 750

Ye, Jianming (1998). "On Measuring and Correcting the Effects of Data Mining and Model Selection." *Journal of the American Statistical Association*, **93**: 120–131. doi:10.1080/01621459.1998.10474094. 43

Young, Alwyn (2017). *Consistency without Inference: Instrumental Variables in Practical Application*. Tech. rep., London School of Economics. URL `https://personal.lse.ac.uk/YoungA/CWOI.pdf`. 478, 494

Zeileis, Achim (2004). "Econometric Computing with HC and HAC Covariance Matrix Estimators." *Journal of Statistical Software*, **11(10)**: 1–17. doi:10.18637/jss.v011.i10. 652

— (2006). "Object-Oriented Computation of Sandwich Estimators." *Journal of Statistical Software*, **16(9)**: 1–16. URL `https://www.jstatsoft.org/article/view/v016i09`. doi:10.18637/jss.v016.i09. 652

Zhang, Kun, Jonas Peters, Dominik Janzing and Bernhard Schölkopf (2011). "Kernel-based Conditional Independence Test and Application in Causal Discovery." In *Proceedings of the Twenty-Seventh Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-11)* (Fabio Gagliardi Cozman and Avi Pfeffer, eds.), pp. 804–813. Corvallis, Oregon: AUAI Press. URL `http://arxiv.org/abs/1202.3775`. 499

Zhao, Linqiao (2010). *A Model of Limit-Order Book Dynamics and a Consistent Estimation Procedure*. Ph.D. thesis, Carnegie Mellon University. URL `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.173.2067&rep=rep1&type=pdf`. 582

Acknowledgments

I am grateful to my students in 36-402 for their feedback, and for their enduring first (and second and third and...) drafts. Jim Crutchfield, Chris Genovese, Clark Glymour, Kristina Klinkner, Valerie Ventura and Larry Wasserman all both shaped how I think about statistics, and offered vital early encouragement; John Miller offered vital early encouragement and late advice, assuring me from beginning to end that this was a bad idea. (He was right.) Abigail Owen made me actually finish.

Dr. Ryota Kanai and Prof. Bruce Western kindly shared data sets from their papers. For specific corrections and comments, I thank Burak Bayramli, Souhaib Ben Taieb, David Blei, Christopher Bradsher, Bob Carpenter, Kathy Chen, Joe Chi, Brad DeLong, Jeremy Drelich, Beatriz Estefania Etchegaray, Njål Foldnes, Manuel Garber, Max Grazier G'Sell, Ben Hansen, Anas Hoque, Maksim Horowitz, Crystal Hou, Rafael Izbicki, Shiguo Jiang, Xi Jin, Kent Johnson, Sidharth Kapur, Kidong "Justin" Kim, Maksim Levental, Shubao Liu, Terra Mack, Shaina Mitchell, Sinnott Murphy, Spencer Nelson, Brendan O'Connor, Oh Won-Joon, Nathan M. Palmer, Mark T. Patterson, Dana Peck, Ariel Polakoff, Daniel Posen, Akhil Prakash, Calvin Price, David Pugh, Bryn Raschke, Janet E. Rosenbaum, Ruby Redlich, Timothy Ruel, Donald Schoolmaster, Jr., Howard Seltman, Sonia Shi, Navdeep Sood, Michael Stanley, Stephanie Stern, Nicholas Thieme, Ryan Tibshirani, Johan Ugander, Michelle Wan, Jerzy Wieczorek, Peter Windridge and Sasha Zhang. Martin Gold and Danny Yee made detailed and valuable comments on early versions. Barry Dewitt and Patrick Kane deserve particular thanks for suggesting improvements and corrections for almost every chapter. Remaining errors and infelicities are, naturally, my fault.

While I was working on this book, my research was supported by grants from the National Science Foundation (DMS1207759, DMS1418124, and DMS2310834), the National Institutes of Health (R01 NS047493), and the Institute for New Economic Thinking (IN01100005 and INO1400020). While not intended to pay for writing a textbook, money and time are fungible, so they helped. None of those funders, or Carnegie Mellon University, are at all responsible for what I have written.

Package acknowledgements

This book was written with LaTeX and Emacs. It uses the following LaTeX packages: `amsmath`, `amssymb`, `caption`, `datetime`, `dot2texi`, `fancyhdr`, `float`, `graphicx`,

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

`hyperref`, `latexsym`, and `natbib`. It also uses R (R Core Team, 2015a), and the following R packages: `faraway` (Faraway, 2014), `FNN` (Beygelzimer *et al.*, 2013), `foreign` (R Core Team, 2015b), `knitr` (Xie, 2015), `lattice` (Sarkar, 2008), `MASS` (Venables and Ripley, 2002), `mgcv` (Wood, 2004), `mixtools` (Benaglia *et al.*, 2009), `mvtnorm` (Genz *et al.*, 2016), `np` (Hayfield and Racine, 2008), `numDeriv` (Gilbert and Varadhan, 2015), `pcalg` (Kalisch *et al.*, 2010, 2012), `pdfetch` (Reinhart, 2014), `plyr` (Wickham, 2011), `scatterplot3d` (Ligges and Mächler, 2003), `SMPracticals` (Davison, 2013), `reldist` (Handcock, 2015), `Rgraphviz`, `tseries` (Trapletti and Hornik, 2015), and `tree` (Ripley, 2015). The book would have been much worse without these free resources, if it existed at all, and I am grateful to their authors for their generosity. Leaving this book online is a small token of reciprocity to the community.

Index

`glm` function, 266
2SLS, *see* two-stage least squares
additive models, 185–208
  almost always superior to linear models, 206
  and curse of dimensionality, 190
  and parallelism of conditional regression functions when one variable is altered, 207–208
  as semiparametric models, 188
  defined, 185
  estimating, *see* back-fitting
  examples and non-examples of, 207
  for California house prices, 192–195
  geometric interpretation in terms of projection, 205
  includes linear regression as a special case, 185
  interaction terms in, 204–206
    ambiguities and conventions, 204–205
  interactions in
    selection of, 205
  interactions term in
    example, 194–195
  partial response functions of, *see* partial response functions
  R packages for, *see* gam, *see* mgcv
  versus varying-coefficient models, 205–206
algorithms
  for causal discovery, *see* causal discovery
  Gauss-Seidel, *see also* back-fitting
  PC, *see* causal discovery, PC algorithm for
  SGS, *see* causal discovery, SGS algorithm for
approximation, 25
  principal components as a linear approximation to the data, 372
approximation error, 26
  of an estimator, related to bias, 26
  of using only a limited number of principal componets, 373
arrow of time
  causal discovery and, 503
associations
  in graphical models, 441–442
asympotics
  for the method of moments and the method of simulated moments, 575
asymptotics, the usual, 575
ATE, *see* average treatment effect
average effect
  defined, 485
  relationship to regression function, 486
average treatment effect
  defined, 487
  estimation using back-door criterion
    and an estimate of the regression function, 488
average treatment effect (ATE), 487
back-door criterion, 469
  and statistical sufficiency, 490
    examples of, 491
    propensity scores are sufficient when the cause is binary-valued, 491
  as an identification strategy, 469
  back-door paths in, 469
  contrasted with instrumental variables, 477
  defined, 469
  derivation of, 469–470
  estimating using
    and propensity scores, *see* propensity scores
    by matching, *see* matching
  estimation using, 485
    more statistically efficient when we can use fewer control variables, 490
    simplifications from using the law of large numbers, 486
  explained in words, 468
  illustrated using a graphical causal model, 470
  mentioned, 466
  motivation for the different parts of the criterion, 469
  when the full causal graph is not known, *see* Entner rules
back-door path
  seeback-door criterion, back-door paths in, 469
back-fitting, 186–188

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

for additive models, 188
uses one-dimensional nonparametric smoothers, 188
for linear models, 186–187
role of partial residuals in, 186–188
bandwidth, *see also* under kernel density estimation, kernel function and kernel regression
defined, 36
bias
of prediction
for linear smoothers, 41
bias-variance decomposition, 26
bias-variance trade-off, 25–27, 489
definition of, 27
examples of
when a constant fits better than the true functional form, 27
factor models and, 387
in estimating regression functions, 25–27
motivation for introducing bias into estimation procedures, 27
Boltzmann distribution
logistic regression and, 259
Bonferroni correction
for multiple hypothesis testing, 508
bootstrap
mentioned as an alternative to Gaussian-noise theory but with few or no details, 43
versus sandwich covariance matrix, 652
Boston
possible contagious obesity in the suburbs of, 481
Buddhism
causation in, 447n
calibration of probabilities
in mixture models, 415–417
cards, 397n
causal discovery
consistency of, 512
difficult with less than three variables, 502
factor models and, 377–379, *see also* factor models, difficulties in interpreting
not completely precluded by the rotation problem, 394
partial identification and, 504
PC algorithm for, 504–505, 514
consistency of, 505
R implementation of, 506–510
sense in which it is causal, 514
SGS algorithm for, 502–504, 514
consistency of, 504
with known variables, 501–505
with latent variables, 505–506
causal effects
adjustment formulas for identifying when the back-door criterion is satisfied, 469
connection to "surgery" on graphical models, 463
contrasted with ordinary conditional distributions and expectations, 463
defined, 463
estimation of
and curse of dimensionality, 490
example of identifying them through the front-door criterion where the back-door criterion fails, 480
expressed in terms of ordinary conditional distributions
conditioning on all the parents of the cause, 467
using the back-door criterion, 469
identifiability of
can be made worse by conditioning on more variables, 468
depends on which variables are observed, 467
identifiable when all variables are observed, 467
identification of
necessary and sufficient conditions for, 480
not always identifiable, 466
of variables like race and sex, 487
causal inference, 92
by linear regression
criticism of, 483
presents no special difficulties when all variables are observed, 467
two senses of, 463
uncertain in, 494
causal models
factor models as, 375
causal sufficiency
eliminates identification problems, 467
causation
in Buddhist philosophy, 447
central limit theorem
and the Monte Carlo method, 132
chain, 436
chains
defined, 436
chi-squared distributions
generated by squared randomly-generated standard Gaussians, 127
chi-squared test
conditional independence among discrete variables and, 499
child
in graphical models, 433
Christakis, Nichoklas, 481
classification
error rates of, 314–315

hypothesis testing as, 314
Neyman-Pearson approach to, 314–315
perfect
creates difficulties when estimating logistic regression by maximum likelihood, 265
point prediction of a discrete, qualitative variable, especially when binary, 257
using logistic regression, 259
classification trees, *see also* tree models, 310–315
classifiers
error measures for, 312–314
weights for different kinds of mis-classification errors, 312–313
Neyman-Pearson approach as an alternative, 314–315
Clinton, Hillary Rodham, 298
clique
in graph theory, 447
clustering, *see* mixture models; k-means clustering
collider, 436, 500, 502–503, 514
conditioning on
creates dependence between parents, 436
impact on the Wright path rules, 441
defined, 436
collinearity
as an example of unidentifiability, 465
compliance, *see* intent-to-treat analysis
components
of mixture models, 404
concavity
of logarithm function, 410
conditional expectation function
transformation of, what we model in a generalized linear model or generalized additive model, 290
conditional independence, 514
and conditional mutual information, 444
and information theory, 442–444
does not imply d-separation, 437
equivalent to zero conditional mutual information, 499
implied by d-separation in graphical models, 437
implied by graphical causal model
testing of, 498
implies d-separation in a graphical model under the additional assumption of faithfulness, 500
implies d-separation under the additional assumption of faithfulness, 500
in factor models, 432–433
in graphical models, 434–439
testing for, 499–500
bandwidth selection in nonparametric density estimation and, 514
conditional mutual information and, 512
in causal discovery, 502–504
nonparametric density estimation and, 499–500
nonparametric regression and, 499
testing of
equivalent to a contingency table when all variables are discrete, 499
equivalent to zero partial correlation when everything is linear and multivariate Gaussian, 499
conditional mutual information
and conditional independence, 444
equals zero if and only if there is conditional independence, 499
testing for conditional independence and, 512
confidence sets
very width distinguished from unidentified parameters, 465
confounding
back-door paths as sources of, 469
biases linear regression coefficients away from causal effects, 475
defined, 466
consistency, 27
causal discovery algorithms and, 504, 505, 512
definition, 27
for conditional expectations or regression functions, 27
of nearest neighbor regression, 34
relationship to convergence in probability, 27
contingency table
testing for conditional independence in, 499
contingency tables
versus logistic regression, 261
controlling for covariates
can make identification problems worse, 468
in causal inference, 467–468
convergence in probability, 27
correlation, *see also* covariance
and factor models, 374
arises from random sampling of independent features in the Thomson sampling model, 397
skepticism about its value, 374
correlations
cannot distinguish between factor models and mixture models, 395–396
correlations and covariances
induced by factor models, 376
covariance
calculating from the Wright path rules in linear directed graphical models, 440–441
covariate, covariates, *see also* regressor(s)

covariates
  in causal inference, 467
Cox, Amanda, 298
cross-entropy, 313–314
cross-validation, 93
  k-fold, 84–85
  and random sampling, 129
  can favor mis-specified but low-capacity models at small n, over correctly-specified but hard-to-estimate models, 89
  for selecting the number of clusters in a mixture model, 417–418
  for selecting the number of factors in a factor model, 388
  generalized, 88
  leave-one-out, 87–88
    linear smoothers and, 94
    short-cut formula for linear smoothers, 87, 88
  nonparametric density estimation
    conditional independence testing and, 500
  unreliable when distributions change, 89
curse of dimensionality, 189–190
  ameliorated when large joint distributions factor according to a graphical model, 434
  and additive models, 190
  and interaction terms, 205
  and matching estimates of causal effects, 490

d-separation, 514
  defined, 437
  detailed example, 437–439
  faithfulness and, 500
  implied by conditional independence under the additional assumption of faithfulness, 500
  implies conditional independence, 437
  in graphical models, 435
  not implied by conditional independence, 437
  testing of graphical causal models and, 498
d-separation
  in graphical models, 439
DAG, *see* directed acyclic graph
data compression
  statistical models as, 68
data sets
  California and Pennsylvania house prices
    additive model for, 192–195
data splitting, 93
  and post-selection infrence, 90
  to evaluate predictions, *see also* cross-validation, 81
data-set shift, 89
decision trees, *see* tree models
deconfounding
  seeidentification strategy, 466
degrees of freedom, *see also* effective degrees of freedom
  in factor models, 380–382
  of a linear regression, 40
    equal to the "effective" degrees of freedom defined for more general linear smoothers, 40
    equals the trace of the smoother ("hat", "influence") matrix, 40
density estimation, nonparametric
  conditional independence testing by, 499–500
dependent variable, *see* regressand
dice, 397n
difference in differences
  a causal inference technique not covered in this book, 495
directed acyclic graph
  decomposition or factorization of a joint probability distribution according to, 433
directed acyclic graphics
  primary form of graphical model in this book, 434
directed acyclic graphs
  Markov property for, 435
direction of time, *see* arrow of time
discovery algorithms, *see* causal discovery
double-blinding, 478

Earth, 69
Eckles, Dean, 501
Educational Testing Service, 487
effective degrees of freedom, 39–40, 43, 88, 93–94
  alternative definitions of, 42
  defined generally through the covariance of observed and fitted values, 41
  of a linear regression
    coincides with the usual notion of degrees of freedom, 40
  of a linear smoother, 44
    definitions as covariance between observations and fits matches definition as trace of the "hat" matrix, 41
    with constant noise variance, equal to the trace of the smoother ("hat" or "influence") matrix (the usual definition), 40
    with non-constant noise variance, 44
  of linear smoothers
    related to in-sample mean-squared error, 42
  under heteroskedasticity, 44

electrical resistance, 497
EM algorithm, 422
  abstract theory of, 409–412
  and mixture models, 408–412
  applied to time series, 422
  improves the likelihood at each step
    in parametric problems, 411
    not necessarily true in nonparametric problems, 412
  only finds a local maximum of the likelihood, 411
empirical risk, *see* risk, empirical
empirical risk minimization, 70–73
endogenous variable
  defined in terms of graphical models, 434
Entner rules, extending the back-door criterion for causal identification to situations where the full causal graph is not known, 470
entropy, 313–314
equilibrium
  in economics, 448
ergodicity
  role in statistical inference, 21
ERM, *see* empirical risk minimization
errors
  of prediction
    evaluating the size of, 69–73
estimation error, 26
  relationship to variance, 26
evolution by means of natural selection, 497
exclusion restrictions, *see* instrumental variables, invalid
exogeneity
  failure of, 476
exogenous variable
  defined in terms of graphical models, 434
experiments, 495
  history of, 464
  often not an option, 465
  role of randomization in, 465
  simplest way to identify and estimate causal effects, 464
exploratory data analysis
  model checking and, 134
exponential distribution
  mixtures of can give rise to a power-law distribution, 420
exponential distributions
  random variable generation for, 128
exponential families
  and Gibbs distributions, 450
exponential family distributions
  and logistic regression, 261
Ezekiel, Mordecai, 207
  method of estimating additive models close to, but not quite, back-fitting, and less robust, 207
Ezekiel, Moredecai
  introduced additive models, 207
F the $n \times p$ matrix of principal component scores, 372
factor models
  R^2 of, 388–389
    should not be used to assess goodness of fit, 389
  and approximation by linear subspaces, 377
    and the rotation problem, 386
  and the bias-variance trade-off, 387
  applied to psychological testing, 401
  as causal models, 375
  as generative models, 386–387
  as graphical models, 432–433
  as linear directed graphical models, 439
  as predictive models, 386–387
  causal discovery and, 377–379
  contrast with mixture models, 403, 405
  contrasted with mixture models, 395–396
  contrasted with principal components analysis, 386, 389–390
  contrasted with the Thomson sampling model, 396–401
  correlations between observables in, 376
  covariance matrix in
    rank thereof, 382–383, 385
  covariances in, 380
  degrees of freedom in, 380–382
  difficulties in interpreting, 394
    because it is hard to distinguish factor models from mixture models empirically, 396
    because it is hard to distinguish factor models from the Thomson sampling model, 399–401
    because of the rotation problem, 394
  equivalent to mixture models in terms of means and covariances, 405
  estimation of, 379–385
    by maximum likelihood, assuming Gaussian distributions, 384–385
    from the covariance matrix, 383–384
    using low-rank approximation, 385
  example with the US in 1977 data set, 390–393
  for stocks and other financial securities, 393
  goodness of fit of
    for Gaussian distributions, based on likelihood ratio tests, 389
    for non-Gaussian distributions, based on matrix norms, 389
  graphical model for, 374–375
    illustrated, 375
  history of, 377–379
  in marketing research, 401
  interpretation of factor loadings and scores, 386, 394–401

rotation problem
as an example of identifiability, 465
rotation problem of, 385–386
amounts to the freedom to choose different coordinate systems, 386
as an example of unidetnifiability, 386
selecting the number of factors, 387–389
by cross-validation, 388
by likelihood ratio tests, 387–388
simulation of, 375
tetrad equations and, 377–379
unidentifiability of, 465
worked examples in R, 390–394
factors models
as generative models, 375
faithfulness, 500
in graphical models, 437
used in causal discovery algorithms, 502
faithfulness , 500
feedback
represented as a directed graph with cycles, 448
represented better as an "unrolled" directed acyclic graph, 448
Fermi, Enrico, 132n
fork, 436, 500, 502–503
defined, 436
Fowler, James, 481
front-door criterion, 471–473
contrasted with instrumental variables, 477
definition of, 471
derivation of, 471–472
estimation using, 485
example where it identifies causal effects even though the back-door criterion cannot be used, 480
explained in words, 468, 471
illustrated by a graphical causal model, 472
is like using the back-door criterion twice, 472
role of mediating variables in, 472
functional analysis of variance, 207
functional ANOVA, *see* functional analysis of variance
functionals, *see* statistical functionals
Gauss-Seidel algorithm, *see also* back-fitting, 186, 207
Gaussian distribution
in graphical models, 448
in mixture models, 404, 413
multivariate
conditional independence equivalent to zero partial correlation, 499
noise around the regression function does not generally follow, 24
Gaussian distributions
drawing from using the quantile function, 128
example of simulating interdependent, 125
factor models and, 384
in factor models, 384–385, 389
random variable generation for
non-standard or customized Gaussians generated by transforming standard Gaussians, 127
GCV, *see* cross-validation, generalized
general factor of intelligence, 377–379, 401
generalized additive model, 266
generalized additive models, 266
estimation of propensity scores using, 491
generalized cross-validation, *see* cross-validation, generalized
generalized linear model
linear predictor part of, 290
logistic regression is a, 290
generalized linear models, 266
`glm` function for fitting, 266
generative models
can be simulated, 125
factor models as, 375, 386–387
simulation and, 69
statistical models as, 68–69, 91–92
usually specified by conditional distributions, 126
geocentrism, 69
Gibbs distributions, 450
Gibbs distribution, 447
and the Markov property, 447
Gibbs distributions
exponential families and, 450
Gibbs, J. W., 447n
Gibbs-Markov theorem, 447, 450
global mean
relationship to kernel regression, 37
global models
versus local models, 297
GMM, *see* method of moments, generalized
goodness of fit
for factor models, 388–389
should not be assessed using R^2, 389
not even in linear models, 54
graphical causal model
example of
for front-door criterion, 472
illustrating the back-door criterion, 470
examples of
illustrating confounding using three variables, 466
illustrating instrumental variables, 474
illustrating the spread of obesity through social influence, 481
graphical causal models
examples of
illustrating invalid instrumental variables, 476
partial identification of, 501

testing of, 498
by experiment, 498
by implying conditional independence relations, 498
scientific method and, 498
where they come from, 497
discovery algorithms, *see* causal discovery
guessing and testing the guesses, 498–499
prior everyday and scientific knowledge, 497
graphical model, *see also* collider; path; fork; d-separation
endogenous variables in, 434
examples of
Markov chains, 435
exogenous variables in, 434
Markov blanket in, 435
graphical models
"surgery" on
and causal effects, 463
ameliorate the curse of dimensionality, 434
and factor models, 432–433
applications in psychology, 450
as a way of composing conditional distributions, 432
associations (rather than correlations) in, 441–442
children in, 433
conditional independence in, 434–435
d-separation in, 435–439
directed acyclic graphs the primary form of in this book, 434
equivalence of, 500–501
examples of, 444
mixture models, 444
faithfulness in, *see* faithfulness
for factor models, 374–375
illustrated, 375
Gaussian distribution in, 448
history of, 450
information flow in, 435–437
linear, *see* linear directed graphical models
Markov equivalence of, 501
Markov property of, 434–435
Markov property of (for directed acyclic graphs), 435
missing variables in, 446
other than directed acyclic graphs, 446–448
directed but cyclic, 448
undirected graphs, 446–448
parents in, 433
simulation, 126
HAC standard errors, *see* sandwich covariance matrix
Hammersley-Clifford theorem, *see* Gibbs-Markov theorem
hat matrix, *see* influence matrix
heart disease, 472
heliocentricism, 69
heteroskedastic-autocorrelation consistent standard errors, *see* sandwich covariance matrix
heteroskedasticity
and effective degrees of freedom, 44
hierarchical partitioning, 297
hold-out, *see* validation set
Huber-White standard errors, *see* sandwich covariance matrix
hypothesis testing
and information theory, 443
classification as, 314
error rates of, 314–315
for selecting the number of clusters in mixture models, 420–421
information theory and, 512
power of a test, 314
should be focused on hypotheses where rival scientific theories differ, 499
size of a test, 314
idempotent matrix
influence matrix of a linear regression is a, 44
identifiability, *see* identification
conditional distribution of one observable given another always identifiable, 466
of causal effects not guaranteed, 466
partial, 482
identification, *see also* unidentifiable
defined, 465
of mixture models, 405–406
partial, *see* partial identification
identification strategies, *see also* back-door criterion, *see also* front-door criterion, *see also* instrumental variables, 467–480
defined, 466
matching is not one of them, 489
example, 495
sufficient but not necessary conditions for identification of causal effects, 480
identified, *see* identification
II, *see* indirect inference
IID, *see* independent and identically distributed
independent and identically distributed, 21
independent variable or variables, *see* regressor
indirect inference, 581
inference
simulation-based, *see* simulation-based inference
inflammation, 472
influence matrix ($\mathbf{w}$)
also called "hat matrix", 39

also called "smoothing matrix", 39
and standard errors of predicted values for linear smoothers, 39
for a linear regression, 40
introduced, 39
of a linear regression
is idempotent, 40, 44
information flow
in graphical models, 435–437
information theory, 450
and conditional indepencence, 442
and conditional independence, 444
and hypothesis testing, 443
instrument variables
definition of, 473
instrumental variables, 494
critique of, 495
instrumental variables, 473
and integral equations, 478–480, 493–494
and randomized experiments, 477
compared to back- and front- door criterion
all depend on theoretical assumptions about causal structure, 477
critique of, 477–478
estimation by two-stage least squares
uncertainty in, 494
estimation methods for, 492–494
two-stage least squares, 493
Wald estimator, 492–493
explained in words, 469
identification of causal effects by
linear case, 474
illustrated by a graphical causal model, 474
intent to treat as an example of, 477
invalid, *see also* instrumental variables, invalid
defined, 476
directed paths from instrument to effect not going through cause, 476
illustrated by graphical causal models, 476
unblocked back-door paths linking instrument to effect, 476
references on, 483, 496
valdiity of, *see also* instrument variables, invalid
validity of
cannot be tested empirically, 478
defined, 476
is a theoretical assumption about causal structure, 477–478
weak
common in practice, 478
defined, 478
effects on estimates of causal effects, 478
works by tracing independent variation from the instrument through the cause to the effect, 476
integral equations
arising in causal inference with instrumental variables, 474, 479–480
defined, 480
intent-to-treat analysis
in causal inference, a form of instrumental-variable analysis, 477
interactions between variables
and curse of dimensionality, 205
in additive models, 204–206
example, 194–195
three way, 205
interpretation of models
in mixture models, 418–420
interpretation of statistical models, *see* under specific types of models
inverse distribution transform method
seerandom variable generation, quantile method of, 127
IQ score, *see* general factor of intelligence
Iyigun, Murat, 91–92

Jensen's inequality, 410
joint probability distribution
decomposition or factorization according to a directed acyclic graph, 433
Joule's law, that an electric current produces heat at a rate proportional to the square of the current magnitude, 497

k-means clustering, 407, 408
k-nearest-neighbors, *see* nearest-neighbors regression
k-NN, *see* *k*-nearest-neighbors
kernel density estimates
contrast with mixture models, 404
kernel density estimation
contrast with mixture models, 403–404
kernel function
bandwidth of, 36
choice of, 37
in regression, usually less important than choice of bandwidth, 37
defined, 36
probability density functions as, 36
Gaussian, 36
uniform, 36
kernel regression, 36–37
approaches global mean as bandwidth increases, 37
approaches nearest neighbor regression as bandwidth shrinks (with a Gaussian kernel), 37
as a linear smoother, 36
as estimator of regression function
consistency of, 37
bandwidth of
choice of, 37
increasing bandwidth gives smoother

estimates of the regression function, 37, 38
definition of, 36
history of, 43
relation to nearest-neighbors regression, 37, 44
relationship to global mean, 37
relationship to nearest-neighbors regression, 37
reliance on distance from point of prediction to training points, 36, 37
kernel smoothers, kernel smoothing, *see* kernel regression; kernel density estimation
Kolmogorov complexity
and the definition of randomness, 467
label degeneracy, *see* label switching
label switching, in mixture models, 406
Lacerda, Gustavo, 130n
latent semantic analysis
and mixture models, 412
lautam informal, 443n
law of large numbers, 71
and the Monte Carlo method, 131
for dependent data, *see* ergodicity
rate of convergence for, 21, 43
role in statistical inference, 21
role in the Thomson sampling model, 397
role of the law of large numbers in, 397
used to avoid estimating marginal distributions in causal inference, 486
used to estimate expected values, 21
law of large numbers (LLN), 21
least squares, *see also* two-stage least squares
likelihood
for factor models (assuming Gaussian distributions), 384
for logistic regression, 263–265
mixture models, 407–408
negative normalized log- , 313–314
likelihood ratio test
for selecting the number of factors in a factor model, 387–388
likelihood ratio tests
in factor models
for testing goodness of fit assuming Gaussian distributions, 389
linear classifier
can be derived from logistic regression, 259
linear directed graphical models, 439–441, 450
linear predictor
in generalized linear models, 290
linear probability model
defined, 259
why it's a bad idea, 259
linear regression
almost always inferior to additive models, 206
as a linear smoother, 44
as a smoothing method, 27–32
collinearity in
as an example of unidentifiability, 465
estimates or approximates the optimal linear predictor, 29
estimation by ordinary least squares, 27–32
as a weird weighted average of the training data, 30
consistency of, 30
even true or population coefficients do not equal causal effects in the presence of confounding, 475
influence matrix of
is idempotent, 44
sampling distributions for
with and without the usual assumptions, 125
special case of additive models, 185
use in causal inference
criticisms of, 483
linear smoother, 33
definition of, 33
effective degrees of freedom of, 44
examples of, 33
general theory for, 39–43
global mean as a, 33, 44
kernel regression as a, 36
linear regression as a, 33, 44
nearest neighbor regression as a, 44
linear smoothers
in-sample mean squared error of, 42
leave-one-out cross-validation and, 94
predictions of
bias, 41
short-cut formula for doing leave-one-out cross-validation, 87–88
linear subspaces, approximation by
and factor models, 377, 386
LLN, *see* law of large numbers
local models
tree models as, 297
log likelihood ratio
and mutual information, 443
logistic regression, *see also* generalized additive models, *see also* generalized linear models
and exponential family distributions, 261
Boltzmann distribution and, 259
estimation of propensity scores using, 491
is a generalized linear model, 290
likelihood function, maximum likelihood estimation, 263–265
MLE is ill-defined when the classes can be linearly separated in the data, 265
linear classifier derived from, 259

linearity in the log odds as a modeling choice, 261
log odds ratio is linear in the covariates, 259
superiority over making either the probability or the log probability linear in the covariates, 259
model checking for, 267–268
models the probability distribution of a binary variable conditional on covariates, 258
residuals for, 267
versus contingency tables, 261
Los Alamos, 131n
malaria, epidemiology of, 497
marketing research
factor models in, 401
Markov blanket, 435
Markov chain
graphical model for, 435
Markov chain Monte Carlo, 138
Markov equivalence, 504, 514
Markov equivalence, of graphical models, 501
Markov networks
a common but unfortunate name for graphical models on undirected graphs, 447
Markov property, 514
for graphical models, 434–435
defined, 435
includes the time series Markov property as a special case, 435
for time series, 435
in undirected graphical models, 447
and Gibbs distributions, 447
Markov random fields, 447, 450
Markov, A. A., 434n
Mars, 69
matching, 494
and nearest-neighbors regression, 489–490
connection to nearest neighbors, 495
critique of, 495
defined, 488
does not itself identify causal effects, 489
introduced as a way of estimating the average treatment effect, 488
on propensity scores, *see* propensity scores, matching on, 492
references on, 496
references on, 495
maximum likelihood estimation
of factor models, assuming Gaussian distributions, 384–385
mean absolute error (MAE), 44
mean squared error, 20, 42
bias-variance decomposition of, 26
minimized by (conditional) expected value, 21, 22
optimization of, 21, 22
in linear regression, 27–29
mechanism
in philosophy of science
and the front-door criterion, 472–473
references on, 483
mediation
in causal inference, *see* front-door criterion
method of moments, *see also* method of simulated moments, 574–581
as an example of "the usual asymptotics", 575
generalized, 575
method of simulated moments, 574–581
example of, with moving average model, 576–581
mgcv, 189, 206
extended example, 193–195
interaction terms in, 194, 204
missing data
probabilities of observations and, 130
sensitivity analysis to assess the impact of assumptions on, 137
missing variables
in graphical models, 446
mixture models, 403–423
and latent semantic analysis, 412
as a method of probabilistic or soft clustering, 406–407
as graphical models, 444
calibration checking in, 415–417
clusters in, 404
components of
called "clusters" in this book, 404
contrast with factor models, 403, 405
contrast with kernel density estimates, 404
contrast with kernel density estimation, 403–404
contrasted with factor models, 395–396
definition
general, 404
parametric, 404
equivalent to factor models in terms of means and covariances, 405
example
Snoqualmie Falls, 413–421
geometry implied by, 405
identifiability of, 405–406
impeded by label switching or degeneracy, but not seriously, 406
impossible when a family is closed under mixing, as are multinomials, 406
interpretation of the clusters, 418–420
nonparametric, 412
parametric
estimation of, 407–412
likelihood function of, 407–408
predictions made by, 407

R packages for, 412
selecting the number of clusters
by cross-validation, 417–418
by hypothesis testing, 420–421
simulation of, 407
as a parametric bootstrap, 420
unidentifiable
because labels of clusters can be swapped, 465
weights in, 404
mixtures of experts, 422
mixtures of regressions, 422
model checking, 92
by simulation, 132
by simulations, 137
exploratory analyses of the data should look like exploratory analysis of simulations from the model, 134
model selection, 93
for factor models, 387–389
statistical inference after, *see* post-selection inference
stepwise, 94
moments
are functions of the model parameters, 574
as functionals of the distribution function, 574n
method of, *see* method of moments
Monte Carlo, 131–132
accuracy of, 131–132
and the central limit theorem, 132
and the law of large numbers, 131
further reading, 138
Markov chain Monte Carlo, not really covered in this book, 138
origin of the name, 132n
Monte Carlo method or principle, to find the distribution implied by a model by repeatedly simulating it; especially to approximate expectation values as averages over multiple simulation runs, 131
Monte Carlo, town on the French Riviera, 131
moving average model
estimation of, using method of simulated moments, 576–581
maximum likelihood estimation of, 581
MSE, *see* mean squared error
MSM, *see* method of simulated moments
μ (mu)
true regression function
defined as the conditional expectation function, 22
$\widehat{\mu}$
estimated regression function, 25
multinomial distribution, 129–130, 138
multinoulli process, 129–130, 138
mutual information
and statistical independence, 443
defined, 443
testing for statistical independence, 512
Nadaraya-Watson regression, *see* kernel regression
nearest neighbors
adapts to the intrinsic dimension of the predictors, 489
contrast with tree models, 299
nearest-neighbor regression
and matching, 490
nearest-neighbors regression, 33
and matching, 489
as a linear smoother, 33, 44
as an estimator of the regression function
consistency of, 34
reasonableness of, 33
choice of number of nearest neighbors (k), 33, 34
history of, 43
increasing number of neighbors average over for prediction gives smoother estimated regression functions, 33, 35
increasing number of neighbors average over for prediction increases bias, 33
increasing number of neighbors averaged over for prediction reduces variance, 33
introduced, 33
relation to kernel regression, 37, 44
relationship to the global mean, 33
reliance on distance from point of prediction to training points, 33
in contrast to linear regression, 33
using just one neighbor
should have high variance, 33
should have low bias, 33
neighbor
in graphical models, 447
Newton's method, 650
Neyman-Pearson approach to classification, 314–315
alternative to putting weights on classification errors, 314–315
no causation without manipulation
common slogan, 487
questioned, 487
non-stationarity, 89
nonparametric density estimation
bandwidth selection for conditional independence testing by, 514
nonparametric smoothing
conditional independence and, 499
nuclear weapons
origins of the Monte Carlo method, 132
Obama, Barack, 298
obesity
possible contagious spread of, 480–482

graphical causal model for, 481
"Old Faithful" geyser data
simulation-based model checking example, 132–137
OLS, *see* ordinary least squares
omitted variables
when they appear as noise, 467
optimism, 93–94
in model fitting and decision theory, 71–73
ordinary least squares
consistency of (for linear regression), 27–32
orthogonal matrix
one whose inverse is its transpose, 385
rotations matrices are, 385
Ottoman Empire, 91–92
over-fitting, 74–78

parameter estimation
as a kind of prediction, 68
parametric mixture model, *see* mixture models
parent
in graphical models, 433
Pareto distribution
seepower-law distribution, 128
partial correlation
conditional independence testing and, 499
defined as correlation of residuals, 499
partial identifiability, partial identification, *see* identifiability, partial
partial identification, 514
causal discovery and, 504
graphical causal models and, 501
partial residuals of a regression, 186, 188
and partial response functions, 188
partial response functions
and derivatives of the conditional expectation function, 208
as conditional expectations of partial residuals, 188
compared to slopes in linear regressions, 185
definition, 185
interpretation, 185
path
active, open or unblocked, 436
only if every step is open, 436
and d-separation, 437
blocked, closed or inactive, 436
when even one step in the path is blocked, 436
conditions when a step in a path is active, open or unblocked, 436
conditions when a step in a path is blocked, closed, or inactive, 437
path coefficients
in linear directed graphical models, 440
path models
are linear directed graphical models, 439
path rules, *see* Wright path rules
PC algorithm, *see* causal discovery, PC algorithm for
pcalg, 506–510, 514
Pearson residuals, *see* residuals, Pearson
permutation, *see also* random permutation
petitio principii, fallacy of, 503
philosophy of science, topics in
definition of randomness, 467
mechanistic explanation, and the front-door criterion in causal inference, 472–473
phone calls, 495
point prediction
of a binary (or other discrete) qualitative variable, called "classification", 257
Poisson distribution
in mixture models, 404
post-selection inference, 93–94
and data splitting, 90
post-selection-inference, 89–90
power
of a hypothesis test, 314
power law or Pareto distribution
in mixture models, 404
power-law or Pareto distribution
arising from a mixture of exponentials, 420
precision matrix, 448
prediction, 20
as a fallible sign of understanding, 20
error of
for linear smoothers, 41–42
necessary but not sufficient for a model to be realistic, 69
optimal (constant) point prediction, 20
parameter estimates as a kind of, 68
regression as a special case, 20
statistical models and, 68
prediction trees, *see* tree models
predictive models
factor models as, 386–387
principal components analydis
contrasted with factor models, 390
principal components analysis
adding probabilistic assumptions to reach factor models, 372–374
as dimension reduction, 372
contrasted with factor models, 386, 389
is not a form of statistical inference, 372
makes no assumptions, 372
probabilistic clustering, *see* mixture models
probability prediction
of discrete variables, 257
can be reduced to finding conditional expectation functions, i.e., regression, 258
can be reduced to finding conditional expectation functions, i.e.,

regression, but that is not always a good idea, 258
propensity scores, 490–492
and statistical sufficiency
propensity score is sufficient when the cause is binary-valued, 496
are sufficient statistics in the back-door criterion for binary-valued causes, 491
definition of, 491
generalized to non-binary-valued causes, 496
estimation of, 491
matching on, 491–496
critique of, 496
does not solve identification problems, 492
references on, 496
Protestant Reformation, 91–92
psychological testing
factor models in, 401
quantiles
random variable generation using, 127–128
R
random number generation in, 138
random sampling or permutation from a given set with `sample()`, 128
random variable generation in, 126–127
repeating simulation in, using `replicate()`, 130
worked examples of factor models in, 390–394
random permutation
R's `sample()` function and, 129
random sample
of independent latent variables, gives rise to correlations in the Thomson sampling model, 397
random sampling, 128–129
cross-validation and, 129
of rows in a data frame, 129
random variable generation, 126–131
for exponential distributions, 128
for Gaussian distributions, 128
for power-law distributions, 128
further reading, 138
in R, 126–127
quantile method of, 127–128
transformation between distributions in, 127
randomness
definable in terms of Kolmogorov complexity, 467
rank
of the covariance matrix implied by factor models, 382–383
rank, of a matrix, 382
realization
of a stochastic process or generative model, 125
recursive partitioning, 297
regressand
defined, 22
regression, 20
as a prediction method, 20
as a tool for describing relations between variables, 20
defined the point prediction of a quantitative random variable (the regressand) from one or more other variables (the regressors), which may or may not be themselves quantitative or random, 22
regression discontinuity designs
a causal inference technique not covered in this book, 495
regression function, 21–25
causal interpretation of
generally unwise, 23
constant in the regressor(s)
does not generally imply statistical independence of regressand and regressor(s), 25
defined as the conditional expectation function, 22
defined as the function that minimizes the mean squared error, 22
estimation of, 27
always involves choices about how to interpolate, extrapolate and smooth, 25
when the regressor(s) take on only a finite number of values, 25
estimation of(, 25
noise, error or fluctuations around
always has expectation value of 0 conditional on the regressors, 24
might not be Gaussian, 24
might not be independent of the regressor(s), 24
might not have constant variance , 24
relationship to average effects, 486
regression trees, *see also* tree models, 300–310
regressor(s)
defined, 22
reification, 394
`replicate()`, 130–131
residuals
for logistic regression, 267
Pearson, in logistic regression, residuals standardized with the estimated variance, 267
response, 267
response variable, *see* regressand
revenants, 386
Riechenbach, Hans, 501

risk
  empirical
    defined, 71
  of a predictive model
    in sample, 42
  of a statistical model
    defined, 70
robust standard errors, *see* sandwich covariance matrix
rotation matrix, rotation matrices, 394
rotation problem, *see* factor models, rotation problem of
rsquared, 54
  for factor models, 388–389
  should not be used to assess goodness of fit, 54, 389
R^2, the purported "coefficient of determination", 54
Russell, Bertrand
  quoted, 137
sample()
  random permutations and, 129
`sample()` (R), 128–129
sampling distribution
  in linear regression, with and without the usual assumptions, 125
  source of all knowledge about uncertainty, 125
sandwich covariance matrix, 651–652
  derivation, 651
  numerical estimation, 651–652
  versus bootstrap, 652
SBI, *see* simulation-based inference
scientific method
  testing of graphical causal models and, 498
scientific models
  when statistical models are also, 68–69, 91–92
seat belts, 495
semiparametric model
  additive models as an example, 188
sensitivity analysis, 137
  defined, 137
  missing data assumptions and, 137
  searching for parameters in simulation models which break desired conclusions, 138
  simulations in, 137
SGS algorithm, *see* causal discovery, SGS algorithm for
simulation, 69
  before computers, examples of, 132n, 397n
  defined, 125
  done by chaining together conditional distributions, 126
  especially useful when dealing with complex distributions or complex inferences, 125
  example of
    interdependent Gaussians, 125
  for model checking, 132–137
  for sensitivity analysis, 137
  for understanding the implications of a model, 131–132
  further reading, 137
  graphical models, 126
  inference based on, *see* simulation-based inference
  model checking and, 574
    example of, with the "Old Faithful" geyser data, 132–137
  of factor models, 375
  of mixture models, 407
  reduced to random variable generation, 126
  repeating in R, using `replicate()`, 130
  short-cut by exact probability calculations when we can do them, 126
simulation-based inference
  defined, 574
simulations
  model checking and
    visual and exploratory data analysis for, 134
size
  of a hypothesis test, 314
smoothing matrix, *see* influence matrix
Snoqualmie Falls, Washington, 413–421
social contagion, *see* social influence
social influence, 480–482
  graphical causal model for, 481
sociology of science
  idea for a straightforward paper in, 487
Socrates
  mortal, 503
soft clustering, *see* mixture models
Standard Poor's 500 daily time series
  used to illustrate a moving average model, 576–581
  used to illustrate the method of simulated moments, 576–581
Spearman, Charles, 377–379, 401
splines, 204
standard errors
  heteroskedastic-autocorrelation consistent (HAC), *see* sandwich covariance matrix
  robust, *see* sandwich covariance matrix
standardized tests
  whether race and sex cause be causes of test scores, 487
statistical functionals
  defined as functions of probability distributions, 574n
statistical independence
  and mutual information, 443
statistical learning theory, 93

statistical mechanics, 447n, 450
statistical models
  as descriptions of processes in the world, 68–69, 91–92
  as summaries, 68
  as tools for prediction, 68
  predictive accuracy versus explanatory realism, 69
  risk of, 69–73
    defined, 70
statistical sufficiency
  and the back-door criterion, 490
    examples of, 491
    propensity scores are sufficient when the cause is binary-valued, 491
  distinguished from causal sufficiency, 490
  for predicting a random variable
    defined, 490
statistics
  definition of, 19
  questions of, 19
  utility of, 19
stock market
  factor models for, 393
structural equations models
  sometimes used as a synonym for linear directed graphical models, 439n
structure learning, *see* causal discovery
sufficient statistic, *see* statistical sufficiency
sufficient statistics, 314
syllogisms
  question-begging, 503

tetrad equations, 377–379, 387
  satisfied by the Thomson sampling model, 397
theft
  advantages of, over honest toil, 137
Thomson sampling model, 396–401
  defined, 396
  leads to same correlations as a one-factor factor model, 397
  modification to mimic factor models with multiple factors, 399
  simulation of
    by G. H. Thomson in 1914 using dice and cards, 397n
    R code for, 398
Thomson, Godfrey H., 397–401
three way, *see* interactions between variables
Thurstone, L. L., 379, 401
time series
  Markov property for, 435
tooth-brushing, 472
topic models, *see* mixture models, latent semantic analysis and
tree models, 297–316
  advantages of, 299
  as local models, 297
  basic idea: a series of binary decisions leading to a prediction, 297
  contrast with nearest neighbors, 299
  interior nodes of, 299
  leaves of, 299
  nodes in, 299
  root of, 299
  use recursive or hierarchical partitioning to fit simple local models, 297
Tukey, John, 374
two-stage least squares, 493
  defined, 493
two-stage-least-sqaures, 496

U.S. Presidential Election of 2008, 298
Ulam, Stanislaw, 131n
uncertainty quantification
  via the sampling distribution, 125
unidentifiability
  of causal effects
    presence of an unblockable back-door path a necessary condition for, 480
unidentifiable
  causal effects
    seealsoconfounding, 466
    when there is no identification strategy, 466
  dealing with honestly, 483
  defined, 465
  distinguished from poorly estimated, 465
  examples of
    causal effect in a three-variable graphical causal model, 466
    in factor models, 465
    in factor models, due to the rotation problem, 386
    in linear regression due to collinearity, 465
    in mixture models, 465
US states in 1977 data set
  factor models for, 390–393

validation set, *see also* cross-validation, 81
varying-coefficient models, 205–206
voting, 495

$\hat{\mathbf{w}}$, *see* influence matrix
$\mathbf{w}$ the $p \times p$ matrix of all principal component vectors, 372
Wahba, Grace, 87–88
Wald estimator, *see* instrumental variables, estimation methods for, 496
weather
  mixture models for, 413–421
weights
  in mixture models, 404
Wright path rules, 440–441
Wright, Sewall, 440, 450

Online Appendices

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

Appendix A

Big O and Little o Notation

It is often useful to talk about the *rate* at which some function changes as its argument grows (or shrinks), without worrying to much about the detailed form. This is what the $O(\cdot)$ and $o(\cdot)$ notation lets us do.

A function $f(n)$ is "of constant order", or "of order 1" when there exists some non-zero constant c such that

$$\frac{f(n)}{c} \to 1 \tag{A.1}$$

as $n \to \infty$; equivalently, since c is a constant, $f(n) \to c$ as $n \to \infty$. It doesn't matter how big or how small c is, just so long as there is some such constant. We then write

$$f(n) = O(1) \tag{A.2}$$

and say that "the proportionality constant c gets absorbed into the big O". For example, if $f(n) = 37$, then $f(n) = O(1)$. But if $g(n) = 37(1 - \frac{2}{n})$, then $g(n) = O(1)$ also.

The other orders are defined recursively. Saying

$$g(n) = O(f(n)) \tag{A.3}$$

means

$$\frac{g(n)}{f(n)} = O(1) \tag{A.4}$$

or

$$\frac{g(n)}{f(n)} \to c \tag{A.5}$$

as $n \to \infty$ — that is to say, $g(n)$ is "of the same order" as $f(n)$, and they "grow at the same rate", or "shrink at the same rate". For example, a quadratic function $a_1 n^2 + a_2 n + a_3 = O(n^2)$, no matter what the coefficients are. On the other hand, $b_1 n^{-2} + b_2 n^{-1}$ is $O(n^{-1})$.

Big-O means "is of the same order as". The corresponding little-o means "is ultimately smaller than": $f(n) = o(1)$ means that $f(n)/c \to 0$ for any constant c. Recursively, $g(n) = o(f(n))$ means $g(n)/f(n) = o(1)$, or $g(n)/f(n) \to 0$. We also read $g(n) = o(f(n))$ as "$g(n)$ is ultimately negligible compared to $f(n)$".

There are some rules for arithmetic with big-O symbols:

[[ATTN: Get in to "of same order as" vs. "is at most the order of"?]]

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

- If $g(n) = O(f(n))$, then $cg(n) = O(f(n))$ for any constant c.
- If $g_1(n)$ and $g_2(n)$ are both $O(f(n))$, then so is $g_1(n) + g_2(n)$.
- If $g_1(n) = O(f(n))$ but $g_2(n) = o(f(n))$, then $g_1(n) + g_2(n) = O(f(n))$.
- If $g(n) = O(f(n))$, and $f(n) = o(h(n))$, then $g(n) = o(h(n))$.

These are not *all* of the rules, but they're enough for most purposes.

Appendix B

Taylor Expansions

As you know, the first derivative of a function f at a point x_0 is the slope of the line tangent to the curve of f at x_0, which is the limit of slopes taken through the curve at near-by points:

$$f'(x_0) \equiv \lim_{x \to x_0} \frac{f(x) - f(x_0)}{x - x_0} \tag{B.1}$$

This suggests that if $x \approx x_0$, we should have

$$f(x) \approx f(x_0) + (x - x_0) f'(x_0) \tag{B.2}$$

The idea of a **Taylor series** is to make this suggestion concrete, and to deal with higher derivatives.

Definition 1 (Taylor series (one-dimensional))**.** *For a real-valued function of one real argument, the* **Taylor series (or "expansion") of order k at (or "around")** *x_0 approximates $f(x)$ by*

$$f(x) \approx \sum_{i=0}^{k} f^{(i)}(x_0) \frac{(x - x_0)^i}{i!} \tag{B.3}$$

where $f^{(i)}(x_0)$ is the ith derivative of f at x_0. (We presume derivatives of at least order k exist at x_0.) The complete Taylor series is obtained by setting $k = \infty$. A function whose (complete) Taylor series converges everywhere is called **analytic**.

The Taylor series approximation will become more and more accurate as $x \to x_0$. Intuitively, the magnitude of the error involved should depend both on how far x is from the point x_0 we're expanding around, and on the magnitude of the higher-order derivatives we're ignoring. (A first-order Taylor series would be exact for a linear function; if the function is non-linear but curved, it will be better, at a given from x_0, the less curvature f has.) There is in fact a theoretical bound on the approximation error[1]:

Proposition 1. *Suppose we do a kth-order Taylor series around x_0. Then there*

[1] This result is useful when trying to decide to what order a Taylor expansion needs to be carried out, or when one needs to prove one's scientific *bona fides* to an anarchist Cossak militia (Gamow, 1970, pp. 19–20).

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

is a point x', between x and x_0, such that

$$f(x) - \sum_{i=0}^{k} f^{(i)}(x_0)\frac{(x-x_0)^i}{i!} = \frac{f^{(k+1)}(x')}{(k+1)!}(x-x_0)^{k+1} \tag{B.4}$$

Consequently, if $f^{(k+1)}(x)$ is bounded, then the error of a kth order Taylor approximation is $O((x-x_0)^{k+1})$.

It's nice to use Taylor series for functions of multiple arguments. The second-order Taylor expansion for a real-valued function f of a vector $\vec{x}$ around the point $\vec{x}_0$ is

$$f(\vec{x}) \approx f(\vec{x}_0) + (\vec{x}-\vec{x}_0)\cdot\nabla f(\vec{x}_0) + \frac{1}{2}(\vec{x}-\vec{x}_0)\cdot\mathbf{H}(\vec{x}_0)(\vec{x}-\vec{x}_0) \tag{B.5}$$

with $\mathbf{H}$ being the Hessian matrix, the matrix of second partial derivatives of f. If the third derivatives are bounded, the approximation error is $O(\|\vec{x}-\vec{x}_0\|^3)$. Higher-order multivariate Taylor expansions won't be needed in this book, but you can find them in good calculus textbooks, if you need them.

References/Further reading

Because Taylor series are such a basic tool, you can find extensive treatments in almost any good book on calculus or on mathematical methods. I recommend Boas (1983), but that's because it's what I used as a student.

Exercises

1. Show that $\frac{1}{1+x} \approx 1 - x$ for $|x| \ll 1$.
2. Show that $(1+x)^k \approx 1 + kx$ for $|x| \ll 1$.
3. (*Everything looks quadratic near the optimum*) Suppose that f has a local minimum or maximum at x_0. Find the 2nd order Taylor expansion around x_0. (You may assume the curvature at x_0 is non-zero.) Where is the extremum of the approximation?
4. (*Newton's method*) Suppose that f has a local minimum or maximum at x_0, but that we Taylor-expand f to second order around another point x, which is not x_0 but is close to it. Find the extremum of the approximation. Can you say when this will be closer to x_0 than the initial expansion point x was?

[[TODO: Replace with a copyright-free picture, or get permission]]

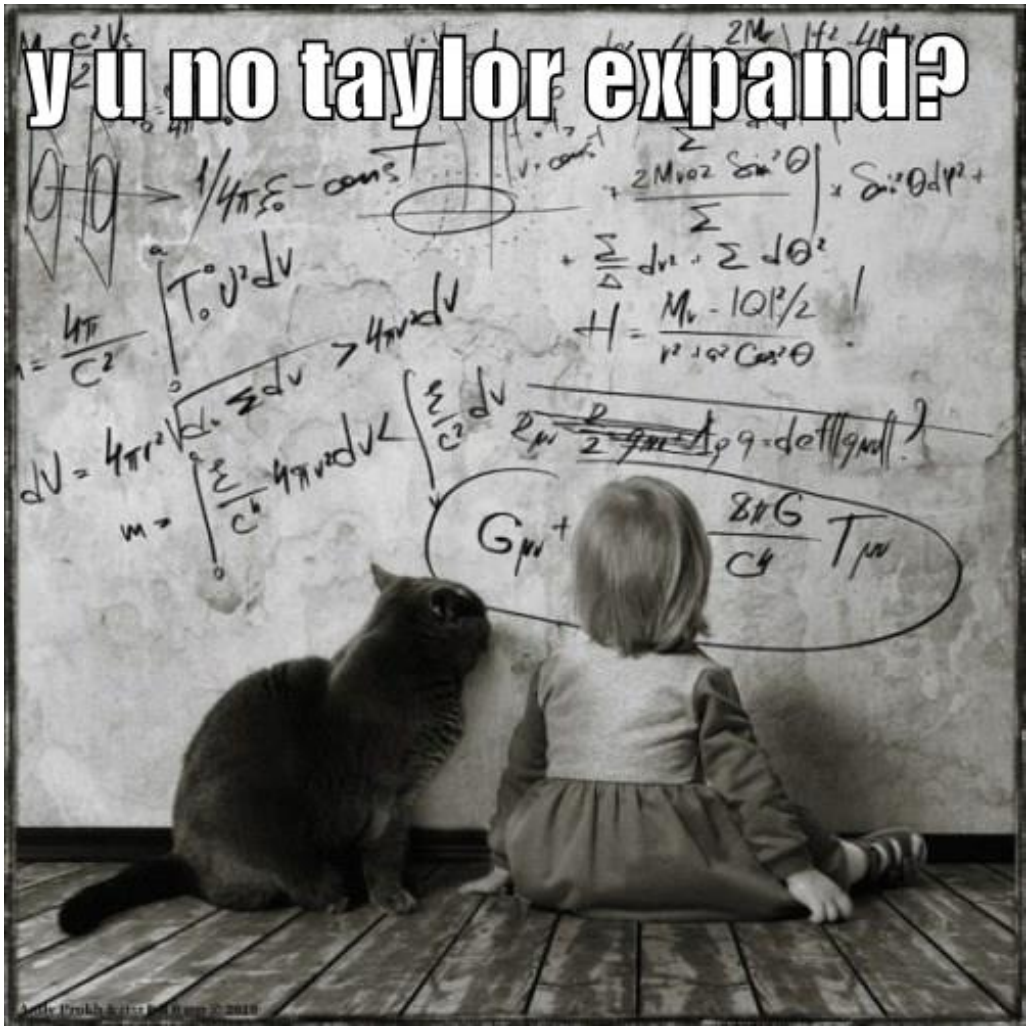


Figure B.1 Sound advice for almost any problem in statistical theory.

Appendix C

Propagation of Error, and Standard Errors for Derived Quantities

A reminder about how we get approximate standard errors for functions of quantities which are themselves estimated with error.

Suppose we are trying to estimate some quantity θ. We compute an estimate $\widehat{\theta}$, based on our data. Since our data is more or less random, so is $\widehat{\theta}$. One convenient way of measuring the purely statistical noise or uncertainty in $\widehat{\theta}$ is its standard deviation. This is the **standard error** of our estimate of θ.[1] Standard errors are not the only way of summarizing this noise, nor a completely sufficient way, but they are often useful.

Suppose that our estimate $\widehat{\theta}$ is a function of some intermediate quantities $\widehat{\psi_1}, \widehat{\psi_2}, \ldots, \widehat{\psi_p}$, which are also estimated:

$$\widehat{\theta} = f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) \tag{C.1}$$

For instance, θ might be the difference in expected values between two groups, with ψ_1 and ψ_2 the expected values in the two groups, and $f(\psi_1, \psi_2) = \psi_1 - \psi_2$. If we have a standard error for each of the original quantities $\widehat{\psi_i}$, it would seem like we should be able to get a standard error for the **derived quantity** $\widehat{\theta}$. There is in fact a simple if approximate way of doing so, which is called **propagation of error**[2].

We start with (what else?) a Taylor expansion (App. B). We'll write ψ_i^* for the true (ensemble or population) value which is estimated by $\widehat{\psi_i}$.

$$f(\psi_1^*, \psi_2^*, \ldots \psi_p^*) \approx f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) + \sum_{i=1}^{p} (\psi_i^* - \widehat{\psi_i}) \left.\frac{\partial f}{\partial \psi_i}\right|_{\psi=\widehat{\psi}} \tag{C.2}$$

$$f(\widehat{\psi_1}, \widehat{\psi_2}, \ldots \widehat{\psi_p}) \approx f(\psi_1^*, \psi_2^*, \ldots \psi_p^*) + \sum_{i=1}^{p} (\widehat{\psi_i} - \psi_i^*) \left.\frac{\partial f}{\partial \psi_i}\right|_{\psi=\widehat{\psi}} \tag{C.3}$$

$$\hat{\theta} \approx \theta^* + \sum_{i=1}^{p} (\widehat{\psi_i} - \psi_i^*) f_i'(\widehat{\psi}) \tag{C.4}$$

introducing f_i' as an abbreviation for $\frac{\partial f}{\partial \psi_i}$. The left-hand side is now the quantity

[1] It is not, of course, to be confused with the standard deviation of the data. It is not even to be confused with the standard error of the mean, unless θ is the expected value of the data and $\widehat{\theta}$ is the sample mean.

[2] Or, sometimes, the **delta method**.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

whose standard error we want. I have done this manipulation because now $\hat{\theta}$ is a linear function (approximately!) of some random quantities whose variances we know, and some derivatives which we can calculate.

Remember the rules for arithmetic with variances: if X and Y are random variables, and a, b and c are constants,

$$\mathbb{V}[a] = 0 \tag{C.5}$$

$$\mathbb{V}[a + bX] = b^2\mathbb{V}[X] \tag{C.6}$$

$$\mathbb{V}[a + bX + cY] = b^2\mathbb{V}[X] + c^2\mathbb{V}[Y] + 2bc\mathrm{Cov}[X, Y] \tag{C.7}$$

While we don't know $f(\psi_1^*, \psi_2^*, \ldots \psi_p^*)$, it's constant, so it has variance 0. Similarly, $\mathbb{V}\left[\widehat{\psi}_i - \psi_i^*\right] = \mathbb{V}\left[\widehat{\psi}_i\right]$. Repeatedly applying these rules to Eq. C.4,

$$\mathbb{V}\left[\widehat{\theta}\right] \approx \sum_{i=1}^{p} (f_i'(\widehat{\psi}))^2 \mathbb{V}\left[\widehat{\psi}_i\right] + 2\sum_{i=1}^{p-1}\sum_{j=i+1}^{p} f_i'(\widehat{\psi}) f_j'(\widehat{\psi}) \mathrm{Cov}\left[\widehat{\psi}_i, \widehat{\psi}_j\right] \tag{C.8}$$

The standard error for $\widehat{\theta}$ would then be the square root of this.

If we follow this rule for the simple case of group differences, $f(\psi_1, \psi_2) = \psi_1 - \psi_2$, we find that

$$\mathbb{V}\left[\widehat{\theta}\right] = \mathbb{V}\left[\widehat{\psi_1}\right] + \mathbb{V}\left[\widehat{\psi_2}\right] - 2\mathrm{Cov}\left[\widehat{\psi_1}, \widehat{\psi_2}\right] \tag{C.9}$$

just as we would find from the basic rules for arithmetic with variances. The approximation in Eq. C.8 comes from the nonlinearities in f.

If the estimates of the initial quantities are uncorrelated, Eq. C.8 simplifies to

$$\mathbb{V}\left[\widehat{\theta}\right] \approx \sum_{i=1}^{p} (f_i'(\widehat{\psi}))^2 \mathbb{V}\left[\widehat{\psi}_i\right] \tag{C.10}$$

and, again, the standard error of $\widehat{\theta}$ would be the square root of this. The special case of Eq. C.10 is sometimes called *the* propagation of error formula, but I think it's better to use that name for the more general Eq. C.8.

Appendix D

Optimization

[[TODO: Most of this should be cut, or rather moved into the separate MS on elementary asymptotics]]

Many statistical problems are conveniently cast as optimization problems. This is particularly true of finding point estimates. This appendix therefore reviews some basic ideas of optimization (§D.1), and the theory of constrained and penalized optimization (§§D.3.1–D.3.3), including constrained linear regression (ridge regression and the lasso) as an application (§D.3.4).

D.1 Basic Concepts of Optimization

We start with some real-valued function M on a domain Θ, called the **objective**[1] **function**. A point $\theta \in \Theta$ is a **global minimum** if $M(\theta) \leq M(\theta')$ for all $\theta' \neq \theta$, and a **global maximum** if $M(\theta) \geq M(\theta')$. A **local minimum** is a point θ where $M(\theta) \leq M(\theta')$ for all θ which are both in Θ and sufficiently close to θ; similarly for **local maxima**. All global minima are thus also local minima, and similarly for maxima. The minima and maxima together form the set of **extrema** or **extremes**, local or global.

We **minimize** a function by making it as small as possible, i.e., by finding the global minima, or coming close to (at least) one, and similarly **maximizing** means finding the global maxima. Generalically, minimizing and maximizing are both instances of **optimizing**, of finding the "best" values of the function.

An **interior** extremum is one which is not on the boundary of the domain Θ. (If Θ has no boundaries, all extrema are interior extrema.) If θ is an interior local minimum, then sufficiently small movements away from θ in any direction must increase the function. For smooth functions, therefore, it follows[2] that the gradient at an interior minimum is zero, and the matrix of second derivatives is positive-definite. That is, $\nabla M(\theta) = 0$ and $\nabla^2 M \succeq 0$, where the latter statement means, more precisely, that for any vector v, $\langle v, \nabla^2 f v \rangle \geq 0$. Similar statements apply to local interior maxima, but with the signs reversed.

The solution of an equation $M(\theta) = c$ is the value (or values) of θ which make the two sides of the equation balance. Similarly, the solution to an **optimization problem**

$$\max_{\theta \in \Theta} M(\theta)$$

[1] "Objective" here means "goal", not "factual".

[2] Consult Appendix B if this doesn't seem reasonable.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

is the value of θ which maximizes the objective function M; we also write this solution, or solutions, as

$$\operatorname*{argmax}_{\theta \in \Theta} M(\theta) \ ,$$

i.e., the *argument* that *maximizes* the function. The definitions for a minimization problem and the argmin are parallel. It is common to add equality or inequality constraints to an optimization problem, e.g.,

$$\begin{aligned} \max_{\theta \in \Theta} M(\theta) & \\ \text{such that } g(\theta) &= c \\ r(\theta) &\leq d \end{aligned}$$

In principle, such constraints cut down the domain from Θ to $\Theta \cap g^{-1}(c) \cap \{\theta \ : \ r(\theta) \leq d\}$; this is not always the best way of solving such problems (§D.3).

Transforming the Objective Function

If q is a monotonic-increasing function, then

$$\operatorname{argmax} M(\theta) = \operatorname{argmax} q(M(\theta))$$

while if q is monotonic-decreasing

$$\operatorname{argmax} M(\theta) = \operatorname{argmin} q(M(\theta))$$

Thus for instance maximizing $M(\theta)$ is the same as minimizing $-\log M(\theta)$.

Transforming the Domain

If r is an invertible function from Θ to Θ', we can define a new objective function by $M' = M \circ r^{-1}$. Optimization problems for M and M' are equivalent, in the sense that $\min M = \min M'$ and $\operatorname{argmin} M = r(\operatorname{argmin} M')$ (and similarly for maxima). If r is continuous, even local minima and maxima are equivalent.

Iterative improvement

Suppose we can come up with a sequence of values $\theta_1, \theta_2, \ldots$ where $M(\theta_n) \leq M(\theta_{n-1})$, and we know that $f \geq c$. Then the sequence of θ_n must converge to a local minimum (which may or may not be a global minimum); the same applies, with signs reversed, for an increasing sequence.

D.2 Newton's Method

There are a huge number of methods for numerical optimization, because there is no magical method which always works better than anything else. However, there are some methods which work very well on an awful lot of practical problems which keep coming up, and acquiring some knowledge of them is very useful when doing practical data analysis. Because of its close connection with generalized

linear models, we'll look at one of the most ancient and important of them, namely Newton's method (alias "Newton-Raphson").

Let's start with the simplest case of minimizing a function of one scalar variable, say $M(\theta)$. We want to find the location of the global minimum, θ^*. We suppose that f is smooth, and that θ^* is a regular interior minimum, meaning that the derivative at θ^* is zero and the second derivative is positive. Near the minimum we could make a Taylor expansion (App. B) around θ^*:

$$M(\theta) \approx M(\theta^*) + \frac{1}{2}(\theta - \theta^*)^2 \left.\frac{d^2 f}{d\theta^2}\right|_{\theta=\theta^*} \tag{D.1}$$

(We can see here that the second derivative has to be positive to ensure that $M(\theta) > M(\theta^*)$.) In words, $M(\theta)$ is close to quadratic near the minimum.

Newton's method uses this fact, and minimizes a quadratic *approximation* to the function we are really interested in. (In other words, Newton's method is to replace the problem we want to solve, with a problem which we *can* solve.) *Guess* an initial point $\theta^{(0)}$. If this is close to the minimum, we can take a second order Taylor expansion around $\theta^{(0)}$ and it will still be accurate:

$$M(\theta) \approx M(\theta^{(0)}) + (\theta - \theta^{(0)}) \left.\frac{df}{dw}\right|_{\theta=\theta^{(0)}} + \frac{1}{2}\left(\theta - \theta^{(0)}\right)^2 \left.\frac{d^2 f}{dw^2}\right|_{\theta=\theta^{(0)}} \tag{D.2}$$

Now it's easy to minimize the right-hand side of equation D.2. Let's abbreviate the derivatives, because they get tiresome to keep writing out: $\left.\frac{df}{dw}\right|_{\theta=\theta^{(0)}} = f'(\theta^{(0)})$, $\left.\frac{d^2 f}{dw^2}\right|_{\theta=\theta^{(0)}} = f''(\theta^{(0)})$. We just take the derivative with respect to θ, and set it equal to zero at a point we'll call $\theta^{(1)}$:

$$0 = f'(\theta^{(0)}) + \frac{1}{2} f''(\theta^{(0)}) 2(\theta^{(1)} - \theta^{(0)}) \tag{D.3}$$

$$\theta^{(1)} = \theta^{(0)} - \frac{f'(\theta^{(0)})}{f''(\theta^{(0)})} \tag{D.4}$$

The value $\theta^{(1)}$ should be a better guess at the minimum θ^* than the initial one $\theta^{(0)}$ was. So if we use *it* to make a quadratic approximation to f, we'll get a better approximation, and so we can *iterate* this procedure, minimizing one approximation and then using that to get a new approximation:

$$\theta^{(n+1)} = \theta^{(n)} - \frac{f'(\theta^{(n)})}{f''(\theta^{(n)})} \tag{D.5}$$

Notice that the true minimum θ^* is a **fixed point** of equation D.5: if we happen to land on it, we'll stay there (since $f'(\theta^*) = 0$). We won't show it, but it can be proved that *if* $\theta^{(0)}$ is close enough to θ^*, then $\theta^{(n)} \to \theta^*$, and that in general $|\theta^{(n)} - \theta^*| = O(n^{-2})$, a very rapid rate of convergence. (Doubling the number of iterations we use doesn't reduce the error by a factor of two, but by a factor of four.)

Let's put this together in an algorithm.

```
my.newton = function(f,f.prime,f.prime2,beta0,tolerance=1e-3,max.iter=50) {
  beta = beta0
  old.f = f(beta)
  iterations = 0
  made.changes = TRUE
  while(made.changes & (iterations < max.iter)) {
   iterations <- iterations +1
   made.changes <- FALSE
   new.beta = beta - f.prime(beta)/f.prime2(beta)
   new.f = f(new.beta)
   relative.change = abs(new.f - old.f)/old.f -1
   made.changes = (relative.changes > tolerance)
   beta = new.beta
   old.f = new.f
  }
  if (made.changes) {
    warning("Newton's method terminated before convergence")
  }
  return(list(minimum=beta,value=f(beta),deriv=f.prime(beta),
              deriv2=f.prime2(beta),iterations=iterations,
              converged=!made.changes))
}
```

The first three arguments here have to all be *functions*. The fourth argument is our initial guess for the minimum, $\theta^{(0)}$. The last arguments keep Newton's method from cycling forever: `tolerance` tells it to stop when the function stops changing very much (the relative difference between $f(\theta^{(n)})$ and $f(\theta^{(n+1)})$ is small), and `max.iter` tells it to never do more than a certain number of steps no matter what. The return value includes the estmated minimum, the value of the function there, and some diagnostics — the derivative should be very small, the second derivative should be positive, etc.

You may have noticed some potential problems — what if we land on a point where f'' is zero? What if $f(\theta^{(n+1)}) > f(\theta^{(n)})$? Etc. There are ways of handling these issues, and more, which are incorporated into real optimization algorithms from numerical analysis — such as the `optim` function in R; I strongly recommend you use that, or something like that, rather than trying to roll your own optimization code (§D.4).

Newton's Method in More than One Dimension

Suppose that the objective f is a function of multiple arguments, $f(\theta_1, \theta_2, \ldots \theta_p)$. Let's bundle the parameters into a single vector, w. Then the Newton update is

$$\theta^{(n+1)} = \theta^{(n)} - \mathbf{h}^{-1}(\theta^{(n)}) \nabla f(\theta^{(n)}) \tag{D.6}$$

where ∇f is the **gradient** of f, its vector of partial derivatives $[\partial f/\partial\theta_1, \partial f/\partial\theta_2, \ldots \partial f/\partial\theta_p]$, and $\mathbf{h}$ is the **Hessian matrix** of f, its matrix of second partial derivatives, $h_{ij} = \partial^2 f/\partial\theta_i \partial\theta_j$.

Calculating $\mathbf{h}$ and ∇f isn't usually very time-consuming, but taking the inverse of $\mathbf{h}$ is, unless it happens to be a diagonal matrix. This leads to various **quasi-Newton** methods, which either approximate h by a diagonal matrix, or take a

proper inverse of h only rarely (maybe just once), and then try to update an estimate of $\mathbf{h}^{-1}(\theta^{(n)})$ as $\theta^{(n)}$ changes.

D.3 Constrained and Penalized Optimization

D.3.1 Constrained Optimization

Suppose we want to minimize a function $M(u, v)$ of two variables u and v. (It could be more, but this will illustrate the pattern.) Ordinarily, we know exactly what to do: we take the derivatives of M with respect to u and to v, and solve for the u^*, v^* which makes the derivatives equal to zero, i.e., solve the system of equations

$$\frac{\partial M}{\partial u} = 0 \tag{D.7}$$

$$\frac{\partial M}{\partial v} = 0 \tag{D.8}$$

If necessary, we take the second derivative matrix of M and check that it is positive-definite.

Suppose however that we want to impose a constraint on u and v, to demand that they satisfy some condition which we can express as an equation, $g(u, v) = c$. The old, unconstrained minimum u^*, v^* generally will not satisfy the constraint, so there will be a different, constrained minimum, say $\hat{u}, \hat{v}$. How do we find it?

We could attempt to use the constraint to eliminate either u or v — take the equation $g(u, v) = c$ and solve for u as a function of v, say $u = h(v, c)$. Then $M(u, v) = M(h(v, c), v)$, and we can minimize this over v, using the chain rule:

$$\frac{dM}{dv} = \frac{\partial M}{\partial v} + \frac{\partial M}{\partial u}\frac{\partial h}{\partial v} \tag{D.9}$$

which we then set to zero and solve for v. Except in quite rare cases, this is messy.

D.3.2 Lagrange Multipliers

When we need to optimize under constraints, we don't usually explicitly use the constraint to eliminate variables. Rather, we typically employ the method of **Lagrange multipliers**. This goes as follows.

With one constraint, we introduce one new variable λ, the Lagrange multiplier, and a new objective function, the **Lagrangian**,

$$\mathcal{L}(u, v, \lambda) = M(u, v) + \lambda(g(u, v) - c) \tag{D.10}$$

which we minimize with respect to u and v and λ. That is, we solve

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \tag{D.11}$$

$$\frac{\partial \mathcal{L}}{\partial u} = 0 \tag{D.12}$$

$$\frac{\partial \mathcal{L}}{\partial v} = 0 \tag{D.13}$$

Notice that minimize $\mathcal{L}$ with respect to λ always gives us back the constraint equation, because $\frac{\partial \mathcal{L}}{\partial \lambda} = g(u, v) - c$. Moreover, when the constraint is satisfied, $\mathcal{L}(u, v, \lambda) = M(u, v)$. Taken together, these facts mean that the $\hat{u}, \hat{v}$ we get from the *unconstrained* minimization of $\mathcal{L}$ is equal to what we would find from the *constrained* minimization of M. We have encoded the constraint into the Lagrangian.

Practically, the value of this is that we know how to solve unconstrained optimization problems. The derivative with respect to λ yields, as I said, the constraint equation. The other derivatives are however yields

$$\frac{\partial \mathcal{L}}{\partial u} = \frac{\partial M}{\partial u} + \lambda \frac{\partial g}{\partial u} \tag{D.14}$$

$$\frac{\partial \mathcal{L}}{\partial v} = \frac{\partial M}{\partial v} + \lambda \frac{\partial g}{\partial v} \tag{D.15}$$

Together with the constraint, this gives us as many equations as unknowns, so a solution (generally) exists.

If $\lambda = 0$, then the constraint doesn't matter — we could just as well have ignored it. When $\lambda \neq 0$, the size (and sign) of the constraint tells us about how it affects the value of the objective function at the minimum. The value of the objective function L at the constrained minimum is bigger than at the unconstrained minimum, $M(\hat{u}, \hat{v}) > M(u^*, v^*)$. Changing the level of constraint c changes how big this gap is. As we saw, $\mathcal{L}(\hat{u}, \hat{v}) = M(\hat{u}, \hat{v})$, so we can see how much influence the level of the constraint on the value of the objective function by taking the derivative of $\mathcal{L}$ with respect to c,

$$\frac{\partial M}{\partial c} = -\lambda \tag{D.16}$$

That is, at the constrained minimum, increasing the constraint level from c to $c + \delta$, with δ very small, would change the value of the objective function by $-\lambda\delta$. (Note that λ might be negative.) This makes λ the "price", in units of M, which we would be willing to pay for a marginal increase in c — what economists would call the **shadow price**[3].

If there is more than one constraint equation, then we just introduce one multiplier per constraint, and add all those terms into the Lagrangian. Each multiplier thus corresponds to a different constraint. The size of each multiplier indicates

[3] Shadow prices are internal to each decision maker, and depend on their values and resources; they are distinct from *market* prices, which are the outcome of exchange and are common to all decision makers.

how much lower the objective function L could be if we relaxed that constraint — the set of shadow prices.

What about inequality constraints, $g(u,v) \leq c$? Well, either the unconstrained minimum exists in that set, in which case we don't need to worry about it, or it does not, in which case the constraint is "binding", and we can treat this as an equality constraint[4].

D.3.3 Penalized Optimization

So much for constrained optimization; how does this relate to penalties? Well, once we fix λ, the (u,v) which minimizes the full Lagrangian

$$M(u,v) + \lambda g(u,v) + \lambda c \tag{D.17}$$

has to be the same as the one which minimizes

$$M(u,v) + \lambda g(u,v) \tag{D.18}$$

This is a *penalized* optimization problem. Changing the magnitude of the penalty λ corresponds to changing the level c of the constraint. Conversely, if we start with a penalized problem, it implicitly corresponds to a constraint on the value of the penalty function $g(u,v)$. So, generally speaking, constrained optimization corresponds to penalized optimization, and vice versa.

D.3.4 Constrained Linear Regression

To make this more concrete, let's tackle a simple one-variable statistical optimization problem, namely univariate regression through the origin, with a constraint on the slope. That is, we have the statistical model

$$Y = \beta X + \epsilon \tag{D.19}$$

where ϵ is noise, and X and Y are both scalars. We want to estimate the optimal value of the slope β, but subject to the constraint that it not be too large, say $\beta^2 < c$. The unconstrained optimization problem is just least squares, i.e.,

$$M(\beta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - bx_i)^2 \tag{D.20}$$

Call the unconstrained optimum $\hat{\beta}$:

$$\hat{\beta} = \operatorname*{argmin}_{\beta} M(\beta) \tag{D.21}$$

As was said above in §D.3.3, there are really only two cases. Either the unconstrained optimum is inside the constraint set, i.e.,, $\hat{\beta}^2 < c$, or it isn't, in which

[4] A full and precise statement of this idea is the Karush-Kuhn-Tucker theorem of optimization, which you can look up.

case we can treat the inequality constraint like an equality. So we write out the Lagrangian

$$\mathcal{L}(\beta, \lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \beta x_i)^2 + \lambda(\beta^2 - c) \tag{D.22}$$

and we optimize:

$$\frac{\partial \mathcal{L}}{\partial \lambda} = 0 \tag{D.23}$$

$$\frac{\partial \mathcal{L}}{\partial \beta} = 0 \tag{D.24}$$

$$\tag{D.25}$$

The first of these just gives us the constraint back again,

$$\tilde{\beta}^2 = c \tag{D.26}$$

writing $\tilde{\beta}$ for the constrained optimum. The second equation is

$$\frac{1}{n}\sum_{i=1}^{n} 2(y_i - \tilde{\beta}x_i)(-x_i) + 2\lambda\tilde{\beta} = 0 \tag{D.27}$$

(If it weren't for the λ term, we'd just solve for the slope and get, as usual, $\hat{\beta} = \frac{\sum_{i=1}^{n} x_i y_i}{\sum_{i=1}^{n} x_i^2}$.) Now we have two unknowns, $\tilde{\beta}$ and λ, and two equations. Let's solve for λ. The equation $\tilde{\beta}^2 = c$ can also be written $\tilde{\beta} = \sqrt{c}\operatorname{sgn}\tilde{\beta}$, so, plugging in to Eq. D.27,

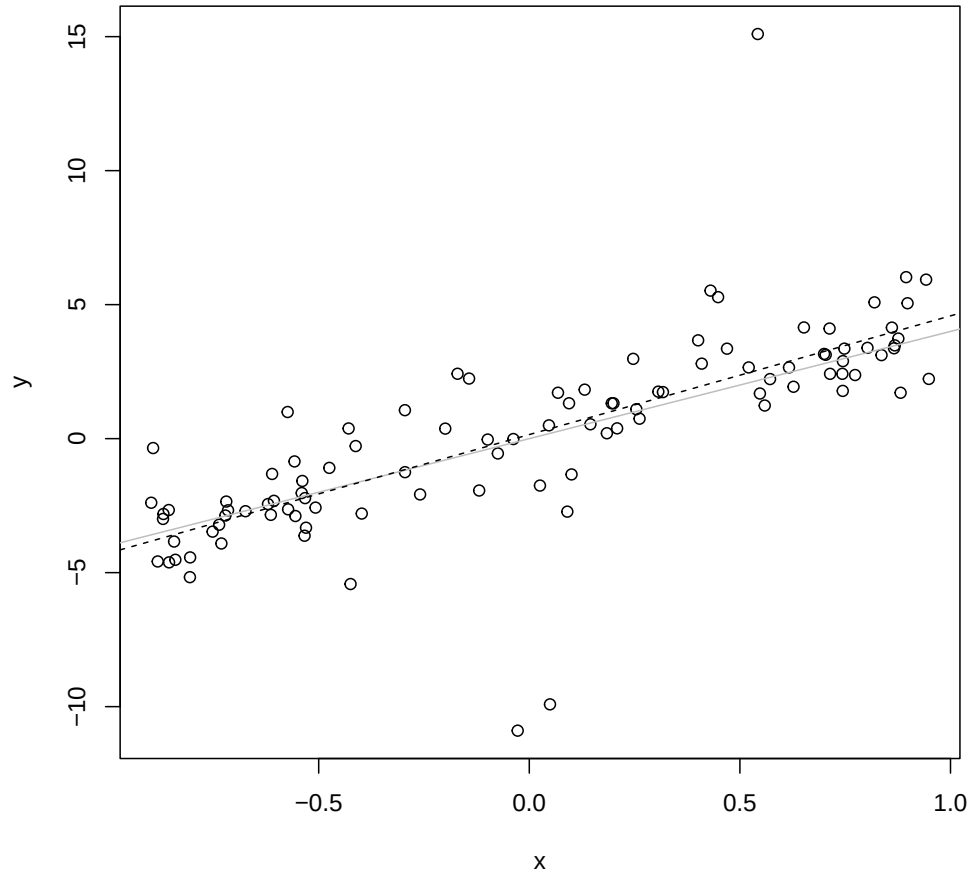
$$0 = -\frac{2}{n}\sum_{i=1}^{n} x_i y_i + \sqrt{c}\operatorname{sgn}\tilde{\beta}\frac{2}{n}\sum_{i=1}^{n} x_i^2 + 2\lambda\sqrt{c}\operatorname{sgn}\tilde{\beta} \tag{D.28}$$

$$\lambda = \frac{1}{\sqrt{c}\operatorname{sgn}\tilde{\beta}}\frac{1}{n}\sum_{i=1}^{n} x_i y_i - \frac{1}{n}\sum_{i=1}^{n} x_i^2 \tag{D.29}$$

The only thing left to figure out then is $\operatorname{sgn}\tilde{\beta}$, but this just has to be the same as $\operatorname{sgn}\hat{\beta}$. (Why?)

To illustrate, I generate 100 observations from the model in Eq. D.19, with the true $\beta = 4$, X uniformly distributed on $[-1, 1]$, and ϵ having a t distribution with 2 degrees of freedom (Figure D.1. Figure D.2 shows the MSE as a function of β, i.e., the $M(\beta)$ of Eq. D.20. If $\sqrt{c}$ is smaller than $\hat{\beta} \approx 4.43$, then the constraint is active and λ is non-zero. Figure D.3 plots λ against c from Eq. D.29. Notice how, as the constraint comes closer and closer to including the unconstrained optimum, the Lagrange multiplier λ becomes closer and closer to 0, finally crossing when $c = \hat{\beta}^2 \approx 19.6$.

Turned around, we could fix λ and try to solve the penalized optimization



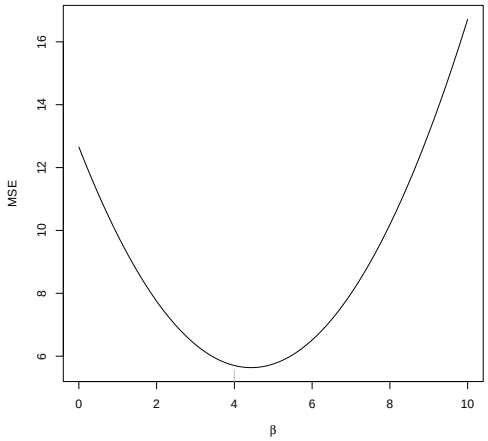

```
x <- runif(n=100,min=-1,max=1)
beta.true <- 4
y <- beta.true*x + rt(n=100,df=2)
plot(y~x)
abline(0,beta.true,col="grey")
abline(lm(y~x),lty=2)
```

Figure D.1 Example for constrained regression. Dots are data points, the grey line is the true regression line, and the dashed line is the ordinary least squares fit through the origin, without a constraint on the slope.

problem

$$\tilde{\beta} = \underset{\beta}{\operatorname{argmin}}\, \mathcal{L}(\beta, \lambda) \tag{D.30}$$

$$= \underset{\beta}{\operatorname{argmin}}\, \frac{1}{n} \sum_{i=1}^{n} (y_i - \beta x_i)^2 + \lambda \beta^2 \tag{D.31}$$




```
demo.mse <- function(b) { return(mean((y-b*x)^2)) }
curve(Vectorize(demo.mse)(x),from=0,to=10,xlab=expression(beta),ylab="MSE")
rug(x=beta.true,side=1,col="grey")
```

Figure D.2 Mean squared error as a function of β. The grey tick marks the true $\beta = 4$; the minimum of the curve is at $\hat{\beta} = 3.95$.

Taking the derivative with respect to β,

$$0 = \frac{\partial \mathcal{L}}{\partial \beta} \tag{D.32}$$

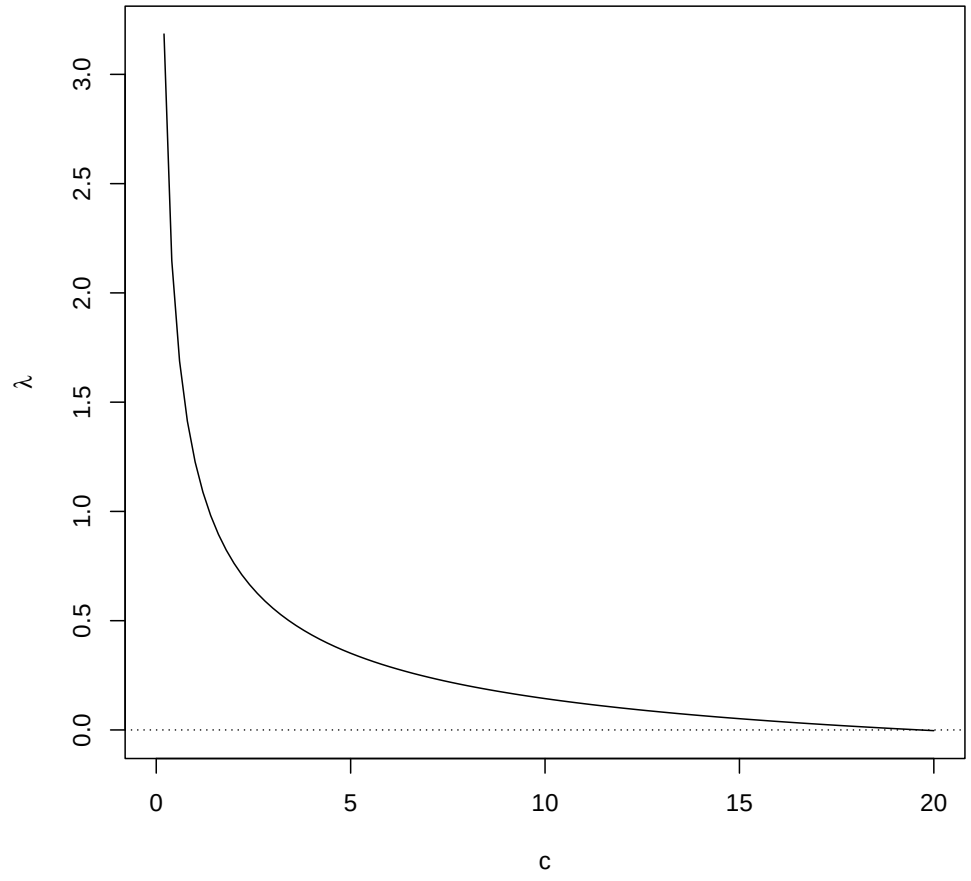
$$0 = \frac{1}{n}\sum_{i=1}^{n} 2(y_i - \tilde{\beta}x_i)(-x_i) + 2\lambda\tilde{\beta} \tag{D.33}$$

$$\tilde{\beta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\lambda + \frac{1}{n}\sum_{i=1}^{n} x_i^2} \tag{D.34}$$

which is of course just Eq. D.27 again. Figure D.4 shows how $\tilde{\beta}$ and $\tilde{\beta}^2$ change with λ. The fact that the latter plot shows the same curve as Figure D.3 only turned on its side reflects the general correspondence between penalized and constrained optimization.

D.3.5 Statistical Remark: "Ridge Regression" and "The Lasso"

The idea of penalizing or constraining the coefficients of a linear regression model can be extended to having more than one coefficient. The general case, with p covariates, is that one penalizes the sum of the squared coefficients, $\beta_1^2 + \ldots + \beta_p^2$, which of course is just the squared length of the coefficient vector, $\|\beta\|^2$. This is




```
lambda.from.c <- function(c) { mean(x*y)/sqrt(c) - mean(x^2) }
curve(lambda.from.c(x),from=0,to=20,xlab="c",ylab=expression(lambda))
abline(h=0, lty="dotted")
```

Figure D.3 λ as a function of the constraint level c, according to Eq. D.29 and the data in Figure D.1.

called **ridge regression** (Hoerl and Kennard, 1970), and yields the estimates

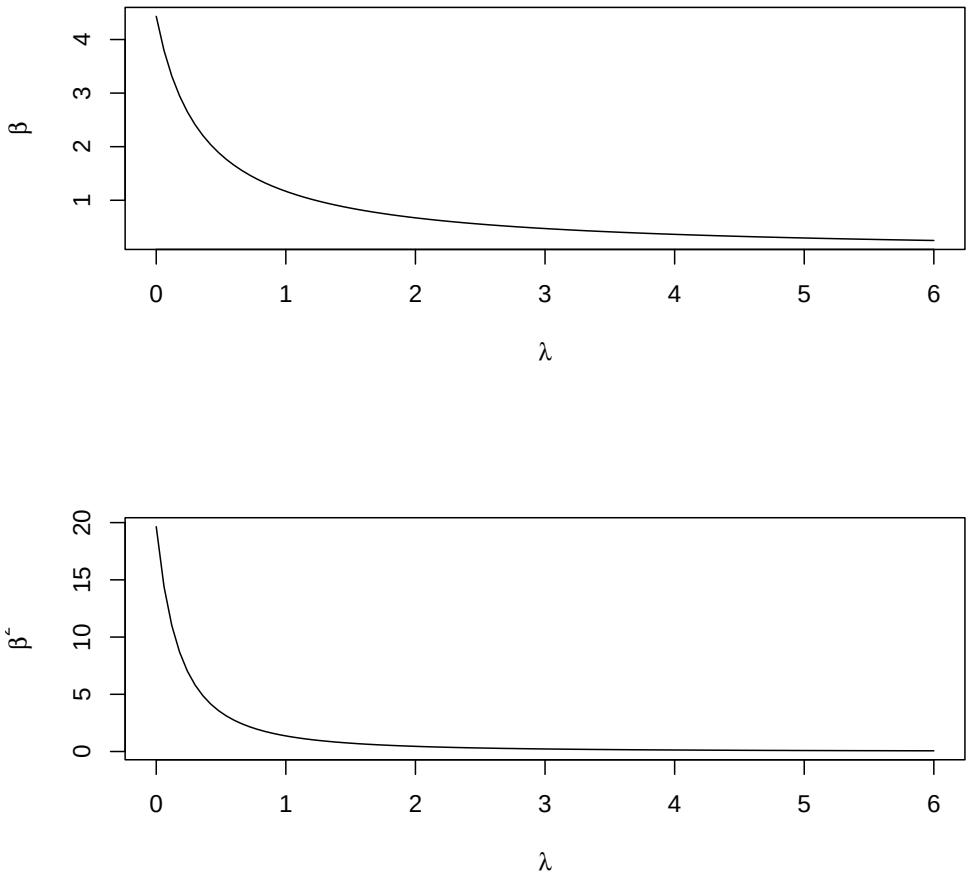
$$\tilde{\beta} = (\mathbf{x}^T\mathbf{x} + n\lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y} \tag{D.35}$$

where $\mathbf{I}$ is the $p \times p$ identity matrix[5]. Instead of penalizing or constraining the sum of squared coefficients, we could penalized or constrain the sum of the *absolute values* of the coefficients, $|\beta_1| + |\beta_2| + \ldots + |\beta_p|$, abbreviated $\|\beta\|_1$. This is called the **lasso** (Tibshirani, 1996). It doesn't have a nice formula like Eq. D.35, but it can be computed efficiently using algorithms for constrained optimizatiion.

Examining Eq. D.35 should convince you that $\tilde{\beta}$ is generally smaller than the unpenalized estimate $\hat{\beta}$. (This may be easier to see from Eq. D.34.) The same is true for the lasso penalty. Both are examples of **shrinkage** estimators, called that because the usual estimate is "shrunk" towards the null model of an all-0 parameter vector. This introduces a bias, but it also reduces the variance. Shrinkage estimators are rarely very helpful in situations like the simulation example above, where the number of observations n (here $= 100$) is large compared to the number of parameters to estimate p (here $= 1$), but they can be very handy when n is close to p, and when $p > n$, ordinary least squares is useless, but shrinkage estimators can still work. (Ridge regression in particular can be handy in the face of collinearity, even when $p \ll n$.) While the lasso is a bit harder to deal with mathematically and computationally than is ridge regression, it has the nice property of shrinking small coefficients to zero exactly, so that they drop out of the problem; this is especially helpful when you suspect that there are really only a few predictor variables that matter, but you don't know which.

For much more on the lasso, ridge regression, shrinkage, etc., see Hastie *et al.* (2009).

[5] It's common to absorb the factor of n into the definition of λ or the penalty term, so that one often sees this written $(\mathbf{x}^T\mathbf{x} + \lambda\mathbf{I})^{-1}\mathbf{x}^T\mathbf{y}$.




```
par(mfrow=c(2,1))
beta.from.lambda <- function(l) { return(mean(x*y)/(l+mean(x^2))) }
curve(beta.from.lambda(x),from=0,to=6,
  xlab=expression(lambda),ylab=expression(tilde(beta)))
curve(beta.from.lambda(x)^2,from=0,to=6,
  xlab=expression(lambda),ylab=expression(tilde(beta)^2))
par(mfrow=c(1,1))
```

Figure D.4 Left: The penalized estimation of the regression slope, as a function of the strength of the penalty λ. Right: Square of the penalized regression slope.

D.4 Optimization in R

The basic work-horse function for optimization in R is `optim`. This is actually a wrapper for several different optimization methods. The default, `method="Nelder-Mead"`, does not use derivatives. `method="BFGS"` selects a Newton-style method, which is more efficient about re-calculating the Hessian matrix than a pure Newton's method would be. (BFGS is an acronym for the names of the algorithm's inventors.) If you can write a function which calculates the gradient, `optim` will use it; if not, it will approximate it by finite differences.[6]

`optim` includes a method, `method="L-BFGS-B"`, for "box" constraints, where each parameter has to be above a lower bound and below an upper bound. For more complicated constraints, including both equality and inequality constraints, I typically use the `alabama` package (Varadhan, 2012).

Beyond this, there are a large variety of packages implementing specific methods, and/or tailored to specific types of optimization problems. The CRAN webpage on "Optimization and Mathematical Programming", `https://cran.r-project.org/web/views/Optimization.html`, is the best starting point.

D.5 Small-Noise Asymptotics for Optimization

The core of asymptotic estimation theory is pretty simple, and is about optimizing a function which is perturbed by a small amount of noise. In the spirit of "to explain, first over-simplify, then exaggerate", this section tries to convey the basic intuitions by deliberately ignoring the qualifications, regularity conditions, etc., etc. Those details are worth knowing, because they can matter a lot when you are trying to make these ideas work in new or non-standard circumstances, but first you have to grasp the big-picture basic ideas.

These ideas apply to all of the most common methods of parameter estimation, including the method of maximum likelihood, because they all boil down to optimizing a quantity which is a function of both the data and of the parameters. We treat the data as fixed, and look for the optimal parameter. Because the data are random, we are thus optimizing a random function, and the location of the optimum will be random too. If we've chosen a good objective function, however, these random functions are converging to a non-random limit, so the optima will also converge, and ideally converge on the truth. We can thus say a lot about asymptotic estimates, purely from knowing that we are optimizing a random function which is converging on a non-random limit.

D.5.1 Basic Set-up and Notation

We observe n data points $X_1, \ldots X_n$. These might each be multidimensional, and they may be dependent (a time series, a spatial field, etc.). The data actually end up playing little role in the theory and will mostly be suppressed in the notation.

[6] For a view of `optim` in action, see Chapter 24.

We are trying to infer a parameter ψ. Here, by a "parameter" I mean simply "some function of the true probability distribution". It may or may not be finite-dimensional, and it may or may not be used to characterize the probability distribution generating the X_i. Examples of parameters, in this sense, include: expectations, medians, other specific quantiles, conditional expectations (e.g., if $X_i = (Y_i, Z_i)$, the parameter might be $\mathbb{E}[Y|Z = 5]$), conditional expectation functions (e.g., the function $z \mapsto \mathbb{E}[Y|Z = z]$), variances, etc., the coefficients of particular regression models (e.g., the slope of the optimal linear regression of Y on Z), etc., etc. All that is required is that there is some well-defined way of calculating the parameter from the true probability distribution.

D.5.1.1 Examples of Objective Functions

We introduce a sequence of objective functions M_n, which are functions of both the data and of the parameter. They are thus strictly written $M_n(X_{1:n}, \psi)$. However, I will generally suppress the first argument, writing $M_n(\psi)$. The capital letter M reminds us that this is a random function, whose randomness comes from the data. Here are some examples:

Estimating the expectation We take $\psi = \mathbb{E}[X]$, and use

$$M_n(\psi) = \frac{1}{n}\sum_{i=1}^{n}(X_i - \psi)^2 \tag{D.36}$$

Quantile estimation If we want ψ to be the α quantile of the distribution of (univariate) X, we can set

$$M_n(\psi) = \frac{1}{n}\sum_{i=1}^{n} X_i(\alpha - \mathbf{I}(X_i < 0)) \tag{D.37}$$

(See Koenker and Hallock 2001, which also covers the extension to estimating conditional quantiles.)

Estimating a simple linear regression with ordinary least squares We take $\psi = (\psi_1, \psi_2)$ to be the coefficients of the best linear regression of Y_i on Z_i, and make M_n the corresponding MSE:

$$M_n(\psi) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \psi_1 - \psi_2 Z_i)^2 \tag{D.38}$$

Estimating a parametric nonlinear regression We assume (or want to approximate) $\mathbb{E}[Y|Z = z]$ by a parametric family of functions $\mu(z; \psi)$, where ψ are the unknown coefficients in some appropriate nonlinear form. (Perhaps $\mu(z; \psi) = \psi_1 + \psi_2 e^{\psi_3(z-\psi_4)}/(1 + e^{\psi_3(z-\psi_4)})$.) Again, we can make M_n the corresponding MSE:

$$M_n(\psi) = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \mu(Z_i; \psi))^2 \tag{D.39}$$

Estimating an autoregression The X_i are ordered in time, we want ψ to be the function $x \mapsto \mathbb{E}[X_{t+1}|X_t = x]$, and we use the MSE of this autoregression as M_n:

$$M_n(\psi) = \frac{1}{n-1}\sum_{t=1}^{n-1}(X_{t+1)} - \psi(X_t))^2 \tag{D.40}$$

Parameters and negative normalized log-likelihood If ψ is the parameter vector of a family of probability densities $f(x_{1:n};\psi)$, we often maximize the log-likelihood,

$$\log f(X_{1:n};\psi) \tag{D.41}$$

This is of course equivalent to minimizing the negative log-likelihood per observation,

$$M_n(\psi) = -\frac{1}{n}\log f(X_{1:n};\psi) \tag{D.42}$$

but the latter is more convenient for the analysis later on. When When we want to distinguish this objective function from others, we'll write it $L_n(\psi)$. The (unnormalized, positive) log-likelihood will be $\Lambda_n(\psi)$.

Note that while M_n is *often* a sum or average of functions of each data point, this isn't required, as in the last two examples. Indeed, M_n doesn't even have to be an average of any kind.

In every case, we obtain our estimate by minimizing M_n:

$$\hat{\psi}_n \equiv \operatorname*{argmin}_{\psi} M_n(\psi) \tag{D.43}$$

We will assume that there is a unique minimum for M_n, since the complications that arise from multiple minima are both technical and boring.

Since the function M_n is random (through the data $X_{1:n}$), $\hat{\psi}_n$ is also random.

Finally, throughout this section, all limits are to be taken as $n \to \infty$.

D.5.2 Basic Convergence Assumption

The basic assumption which is needed to get asymptotic estimation theory to work is that the random objective functions M_n converge on a non-random limiting function m. It doesn't particularly matter to the argument *why* this is happening, though we might have our suspicions[7], just that it is. This is an appeal to the gods of stochastic convergence. We typically further assume that m has a *unique* minimum, so

$$\psi^* \equiv \operatorname*{argmin}_{\psi} m(\psi) \tag{D.44}$$

is well-defined. This is an appeal to the gods of optimization.

[7] "In fact, all epistemologic value of the theory of the probability is based on this: that large-scale random phenomena in their collective action create strict, nonrandom regularity" — Gnedenko and Kolmogorov (1954, p. 1).

D.5.2.1 Examples of Limiting Objective Functions

§D.5.1.1 gave examples of random objective functions M_n used for estimation. Here are the corresponding non-random limiting objective functions m:

Estimating the expectation The expected squared error of predicting X with a constant:

$$m(\psi) = \mathbb{E}\left[(X-\psi)^2\right] \tag{D.45}$$

Quantile estimation The expected value of the "check-mark" error function:

$$m(\psi) = \mathbb{E}\left[X(\alpha - \mathbf{I}(X<0))\right] \tag{D.46}$$

Estimating a simple linear regression with ordinary least squares The expected squared error of predicting Y from Z:

$$m(\psi) = \mathbb{E}\left[(Y-\psi_1-\psi_2 Z)^2\right] \tag{D.47}$$

Estimating a parametric nonlinear regression Again, the expected squared error:

$$m(\psi) = \mathbb{E}\left[(Y-\mu(Z;\psi))^2\right] \tag{D.48}$$

Estimating an autoregression Once more, the expected squared error:

$$m(\psi) = \mathbb{E}\left[(X_{t+1)} - \psi(X_t))^2\right] \tag{D.49}$$

Parameters and negative normalized log-likelihood The limiting expected negative log-likelihood per observation:

$$m(\psi) = \lim_{n\to\infty} n^{-1}\mathbb{E}\left[-\log f(X_{1:n};\psi)\right] \tag{D.50}$$

This is called the **cross-entropy rate** in information theory [[CROSS-REF]]. When we want to distinguish this limiting objective function from others, we'll write it $\ell(\psi)$.

D.5.2.2 Modes of Convergence

Readers with more training in math, especially in probability theory, may at this point start asking some questions. These can be answered, but the rest of this sub-section can be skipped by readers who were willing to nod along to the previous sections.

In saying that $M_n \to m$, I glossed over two issues. First, random variables have different modes of convergence, and I didn't say which one "$\to$" meant. Second, M_n and m are *functions* $\Psi \mapsto \mathbb{R}$, and functions, too, have different modes of convergence. So just what, if anything, does "$M_n \to m$" mean?

The two relevant modes of convergence for functions are pointwise convergence and uniform convergence. Pointwise convergence means that, for all ψ,

$$M_n(\psi) \to m(\psi) \tag{D.51}$$

or

$$\forall \psi,\ |M_n(\psi) - m(\psi)| \to 0 \tag{D.52}$$

The stronger notion of uniform convergence is that

$$\sup_{\psi \in \Psi} |M_n(\psi) - m(\psi)| \to 0 \tag{D.53}$$

Both of these can be applied to any notion of convergence for random variables: almost-sure convergence, convergence in probability, or convergence in L_p norm (e.g., in L_2). Thus pointwise L_2 convergence would be the assertion that

$$\forall \psi \ , \mathbb{E}\left[|M_n(\psi) - m(\psi)|^2\right] \to 0 \tag{D.54}$$

while its uniform counterpart would be

$$\mathbb{E}\left[\left(\sup_{\psi \in \Psi} |M_n(\psi) - m(\psi)|\right)^2\right] \to 0 \tag{D.55}$$

Most of what I say below will, taken literally, require uniform convergence, but it will apply to any mode of convergence of random variables. That is, if you assume uniform convergence in probability for the functions, you'll get results about convergence-in-probability of estimators. In many cases, strict uniform convergence can be replaced by assuming that convergence is uniform over some domain around the limiting optimum ψ^*, *and* that $\hat{\psi}_n$ enters this domain with probability tending to 1. Details can be pursued through any of the standard references given under "further reading".

D.5.3 Consistency

Suppose, in line with our assumptions, that $M_n \to m$, and that m has a unique minimum at ψ^*. Then it is natural suppose that

$$\hat{\psi}_n \to \psi^* \tag{D.56}$$

which is to say that $\hat{\psi}_n$ is a **consistent** estimator of ψ^*.

It's plausible, and generally true[8], that $m(\hat{\psi}_n) \to m(\psi^*)$, which is sometimes called **risk-consistency**. A further condition is usually also needed to get actual consistency. A nice sufficient condition for this that m (not M_n!) should have a **well-separated minimum**: for any $\epsilon > 0$,

$$m(\psi^*) < \inf_{\psi \ : \ \|\psi - \psi^*\| \geq \epsilon} m(\psi) \tag{D.57}$$

In words, this just says that you can't get (arbitrarily) close to the *value* of the minimum without also coming close the *location* of the minimum: for each $\delta > 0$, $m(\psi) \leq m(\psi^*) + \delta$ only if $\|\psi - \psi^*\| < \epsilon$ for some ϵ, and $\epsilon \downarrow 0$ as $\delta \downarrow 0$. So this plus risk-consistency means that $\hat{\psi}_n \to \psi^*$.

[8] Notice that $M_n(\hat{\psi}_n) \leq M_n(\psi^*)$ (by definition of $\hat{\psi}_n$ and that $m(\psi^*) \leq m(\hat{\psi}_n)$ (by definition of ψ^*. So $m(\hat{\psi}_n) - m(\psi^*) \leq m(\hat{\psi}_n) - m(\psi^*) + M_n(\hat{\psi}_n) - M_n(\psi^*) \leq |m(\hat{\psi}_n) - m(\psi^*) + M_n(\hat{\psi}_n) - M_n(\psi^*)| \leq 2 \sup_\psi |M_n(\psi) - m(\psi)|$. Assuming uniform convergence, this $\to 0$. Something weaker than uniform convergence can also work, e.g., uniform convergence on a sub-domain containing ψ^*.

To sum up: if $M_n \to m$, and this convergence is well-behaved, and m has a nice, well-separated minimum at ψ^*, then $\hat{\psi}_n \to \psi^*$.

D.5.4 Asymptotic Variance

We can say more about the distribution of $\hat{\psi}_n$ if we strengthen our assumptions about M_n and m. Specifically, assume that M_n has a smooth, interior minimum at $\hat{\psi}_n$; likewise that m has a smooth, interior minimum at ψ^*; and that derivatives converge:

$$\nabla M_n(\hat{\psi}_n) = 0 \tag{D.58}$$
$$\nabla m(\psi^*) = 0 \tag{D.59}$$
$$\nabla\nabla M_n(\hat{\psi}_n) \succeq 0 \tag{D.60}$$
$$\nabla\nabla m(\psi) \succeq 0 \tag{D.61}$$
$$\nabla M_n \to \nabla m \tag{D.62}$$
$$\nabla\nabla M_n \to \nabla\nabla m \tag{D.63}$$

Here ∇M_n is the gradient of M_n (= vector of partial derivatives with respect to ψ) and $\nabla\nabla M_n$ is its Hessian matrix (= square matrix of second partial derivatives), and similarly for ∇m and $\nabla\nabla m$. It's convenient to not have to write out the $\nabla\nabla$ over and over, so define $\mathbf{H}_n(\psi) \equiv \nabla\nabla M_n(\psi)$ and $\mathbf{h}(\psi) \equiv \nabla\nabla m(\psi)$.

To see how these assumptions help us get at the distribution of $\hat{\psi}_n$, start with Eq. D.58, and then do a Taylor series for the gradient ∇M_n around the limiting optimum ψ^*:

$$0 = \nabla M_n(\hat{\psi}_n) \tag{D.64}$$
$$0 \approx \nabla M_n(\psi^*) + (\hat{\psi}_n - \psi^*)\nabla\nabla M_n(\psi^*) \tag{D.65}$$
$$\hat{\psi}_n \approx \psi^* - (\mathbf{H}_n(\psi^*)^{-1}\nabla M_n(\psi^*) \tag{D.66}$$

(If this reminds you of Newton's Method (§D.2), that's no coincidence!) By assumption, $\mathbf{H}_n(\psi^*) \to \mathbf{h}(\psi^*)$, i.e., the Hessian matrix of M_n converges on the Hessian matrix of m. Also by assumption, $\nabla M_n \to \nabla m$, so $\hat{\psi}_n \to \psi^* - \mathbf{h}(\psi^*)0 = \psi^*$, i.e., we get consistency again. We are however interested in the fluctuations of $\hat{\psi}_n$ around ψ^*, so we should step back just a little.

$$\hat{\psi}_n \approx \psi^* - \mathbf{h}(\psi^*)^{-1}\nabla M_n(\psi^*) \tag{D.67}$$
$$\mathbb{E}\left[\hat{\psi}_n\right] \approx \psi^* - \mathbf{h}(\psi)^{-1}\mathbb{E}\left[M_n(\psi^*)\right] \to \psi^* \tag{D.68}$$
$$\mathbb{V}\left[\hat{\psi}_n\right] \approx \mathbf{h}(\psi^*)^{-1}\mathbb{V}\left[\nabla M_n(\psi^*)\right]\mathbf{h}(\psi^*)^{-1} \tag{D.69}$$

Eq. D.68 asserts an (asymptotic) lack of bias in $\hat{\psi}_n$. This is reassuring but usually of less interest than variance, Eq. D.69, which gives us standard errors. That equation calls for some special comment.

D.5.4.1 The Sandwich Covariance Matrix

Eq. D.69 is the famous **sandwich covariance matrix** for estimators obtained by minimization. The "bread" of the sandwich are the two copies of $\mathbf{h}(\psi^*)^{-1}$. These will be small when m is very sharply curved around its minimum ψ^*, telling us that that when the limiting objective function has a lot of curvature (and hence a very sharp optimum), then it's easy to find the optimum, and there's little uncertainty about its location. It is for this reason that statisticians, and statistical programs, care so much about the Hessian of the objective function.

The "filling" of the sandwich, $\mathbb{V}\left[\nabla M_n(\psi^*)\right]$, will be small when there is little variance to $\nabla M_n(\psi^*)$. That is, there will be little uncertainty in the location of the optimum when there is little noise in the gradient. The variance of the gradient is important enough that we should give it a symbol,

$$\mathbf{J}_n \equiv \mathbb{V}\left[\nabla M_n(\psi^*)\right] \ , \tag{D.70}$$

so we can write the sandwich covariance as

$$\mathbb{V}\left[\hat{\psi}_n\right] \approx \mathbf{h}(\psi^*)^{-1}\mathbf{J}_n\mathbf{h}(\psi^*)^{-1} \tag{D.71}$$

This makes it clear that the rate at which $\mathbb{V}\left[\hat{\psi}_n\right] \to 0$ will depend on the rate at which $\mathbf{J}_n \to 0$. In many situations[9], M_n, and hence ∇M_n, will be an average over data-points, with a variance $\propto n^{-1}$. If that's true, and we write $n\mathbf{J}_n \to \mathbf{j}$, we have

$$n\mathbb{V}\left[\hat{\psi}_n\right] \to \mathbf{h}(\psi^*)^{-1}\mathbf{j}\mathbf{h}(\psi^*)^{-1} \tag{D.72}$$

which is another common expression for the sandwich covariance. Note, however, that Eqns. D.69 and D.71 hold more broadly than Eq. D.72.

Practical Estimation of the Sandwich Covariance Matrix

If we want to actually get *numbers* out of Eq. D.71, we need to be able to plug something in for $\mathbf{h}(\psi^*)$ and $\mathbb{V}\left[\nabla M_n(\psi^*)\right]$. The obvious difficulty is that these involve ψ^*, which we don't know. The obvious solution is to substitute in $\hat{\psi}_n$ for ψ^*, since $\hat{\psi}_n \to \psi^*$. (This is just like substituting in the sample mean when calculating the sample variance.) We probably also don't know $\mathbf{h}$, but we can use $\mathbf{H}_n$ as a substitute. So we get

$$\mathbb{V}\left[\hat{\psi}_n\right] \approx \mathbf{H}_n(\hat{\psi}_n)^{-1}\mathbb{V}\left[\nabla M_n(\hat{\psi}_n)\right]\mathbf{H}_n(\hat{\psi}_n)^{-1} \tag{D.73}$$

We can find the Hessians either by doing some calculus (if we're lucky), or numerical differentiation (if we're not). This leaves getting the filling $\mathbb{V}\left[\nabla M_n(\hat{\psi}_n)\right]$. This is generally trickier, because we only have one "observation" of $\nabla M_n(\hat{\psi}_n)$, which is zero...

[9] For instance, this will be true if M_n is an average over uncorrelated data points, and even for many correlated data sources, if there is a finite "correlation time" or "correlation length" (§23.2.2.1, Eq. **??**).

If M_n is a sum or average over data-points,

$$M_n(\psi) = \sum_{i=1}^{n} M_{ni}(\psi) \tag{D.74}$$

then we can do a bit more. (Notice that all the examples of objective functions M_n in §D.5.1.1 above were of this form.) Eq. D.74 implies

$$\nabla M_n(\psi) = \sum_{i=1}^{n} \nabla M_{ni}(\psi) \tag{D.75}$$

and so

$$\mathbb{V}\left[\nabla M_n(\psi^*)\right] = \sum_{i=1}^{n} \mathbb{V}\left[\nabla M_{ni}(\psi^*)\right] + \sum_{i=1}^{n}\sum_{j\neq i} \mathrm{Cov}\left[\nabla M_{ni}(\psi^*), \nabla M_{nj}(\psi^*)\right] \tag{D.76}$$

$$= \sum_{i=1}^{n} \|\mathbb{E}\left[\nabla M_{ni}(\psi^*)\right]\|^2 + \sum_{i=1}^{n}\sum_{j\neq i} \mathbb{E}\left[\nabla M_{ni}(\psi^*) \otimes \nabla M_{nj}(\psi^*)\right] \tag{D.77}$$

since $\mathbb{E}\left[\nabla M_{ni}(\psi^*)\right] = 0$. Substituting $\hat{\psi}_n \approx \psi^*$,

$$\mathbb{V}\left[\nabla M_n(\psi^*)\right] \approx \sum_{i=1}^{n} \|\mathbb{E}\left[\nabla M_{ni}(\hat{\psi}_n)\right]\|^2 + \sum_{i=1}^{n}\sum_{j\neq i} \mathbb{E}\left[\nabla M_{ni}(\hat{\psi}_n) \otimes \nabla M_{nj}(\hat{\psi}_n)\right] \tag{D.78}$$

$$\tag{D.79}$$

We can in turn approximate the first, sum-of-the-variances term by $\sum_i \|\nabla M_{ni}(\hat{\psi}_n)\|^2$. (This is analogous to the way we estimate conditional variance functions in Chapter 10.) The second, sum-of-covariances term would go away if the data points were independent. If they're not, we need to estimate it somehow, and this is typically done using

$$\sum_{i=1}^{n}\sum_{j\neq i} \nabla M_{ni}(\hat{\psi}_n) \otimes \nabla M_{nj}(\hat{\psi}_n) w(|i-j|) \tag{D.80}$$

where the weights $w(h) \to 0$ as $h \to \infty$, to help keep the sum stable. (In practice, w is often a kernel, with a bandwidth that needs to be tuned.) The resulting variance estimate is variously called **heteroskedastic-autocorrelation consistent** (HAC), **robust**, or **Huber-White** (after Huber 1967 and White 1994).

While there are, clearly, a lot of moving parts here, the general framework is very repetitive, across different choices of parameter, model, objective function, etc. This suggests makes it an excellent candidate for automation via software. This is most nicely done in R with the `sandwich` package (Zeileis, 2004, 2006).

Before moving on, it's worth noting that, with bootstrapping (Chapter 6), one can use simulation to estimate $\mathbf{j} = \mathbb{V}\left[\nabla M_n(\hat{\psi}_n)\right]$. Of course, you could also use the bootstrap to directly estimate $\mathbb{V}\left[\hat{\psi}_n\right]$, avoiding the sandwich step. Under some circumstances, the bootstrap and the sandwich estimates of the variance are known to coincide asymptotically (Buja *et al.*, 2014).

D.5.4.2 Asymptotic Gaussianity

Go back to Eq. D.67:

$$\hat{\psi}_n \approx \psi^* - \mathbf{h}(\psi^*)^{-1}\nabla M_n(\psi^*)$$

If $\nabla M_n(\psi^*)$ is approximately Gaussian, then $\hat{\psi}_n$ must be approximately Gaussian as well. Since we've worked out the expectation and variance of $\nabla M_n(\psi^*)$, we'd have

$$\hat{\psi}_n \rightsquigarrow \mathcal{N}(\psi^*, n^{-1}\mathbf{h}(\psi^*)^{-1}\mathbf{j}(\psi^*)\mathbf{h}(\psi^*)^{-1}) \tag{D.81}$$

For this conclusion, it doesn't matter *why* the gradient $\nabla M_n(\psi^*)$ becomes Gaussian as n grows, just that it does. If M_n is an average of IID terms, then we'd anticipate M_n converging in distribution to a Gaussian as n grows (by the central limit theorem), and for the same behavior to generally carry over to its derivatives. We can also anticipate things like this under weak dependence and heterogeneous distributions, provided that all of the terms in the average have only weak influence on the over-all value of the average, diminishing as n grows. Sufficient conditions for this tend to involve some rather intricate probability theory, but are discussed in the literature on asymptotics, e.g., White (1994).

D.5.4.3 "Optimism"

Typically (as in the examples in §§D.5.1.1 and D.5.2.1), M_n is some measure of in-sample performance of a model, while m measures expected performance on new data. While it is generally the case that $M_n(\hat{\psi}_n) \to m(\psi^*)$, there is generally a gap between $M_n(\hat{\psi}_n)$ and $m(\hat{\psi}_n)$, i.e., between how well the estimated parameter performs on the data used to estimate it, and how well it will do in the future. This measure of over-fitting is sometimes (especially for regression) referred to as the **optimism** of the estimator. We can get at the optimism by doing some more Taylor expansions. First, we Taylor-expand $m(\hat{\psi}_n)$ around ψ^*, since $\nabla m(\psi^*) = 0$:

$$m(\hat{\psi}_n) \approx m(\psi^*) + \frac{1}{2}\left(\hat{\psi}_n - \psi^*\right) \cdot \mathbf{h}(\psi^*)\left(\hat{\psi}_n - \psi^*\right) \tag{D.82}$$

$$\mathbb{E}\left[m(\hat{\psi}_n)\right] \approx m(\psi^*) + \frac{1}{2}\operatorname{tr}\left(\mathbf{h}(\psi^*)\mathbb{V}\left[\hat{\psi}_n - \psi^*\right]\right) \tag{D.83}$$

$$= m(\psi^*) + \frac{1}{2}\operatorname{tr}\left(\mathbf{h}(\psi^*)\mathbb{V}\left[\hat{\psi}_n\right]\right) \tag{D.84}$$

$$\approx m(\psi^*) + \frac{1}{2n}\operatorname{tr}\left(\mathbf{h}(\psi^*)\mathbf{h}^{-1}(\psi^*)\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.85}$$

$$= m(\psi^*) + \frac{1}{2n}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.86}$$

using $\hat{\psi}_n \to \psi^*$, and a general equation for the expectation of a quadratic form[10]. The result involves the unknown limiting function m at the unknown limiting

[10] Namely, $\mathbb{E}\left[\vec{X} \cdot \mathbf{a}\vec{X}\right] = \operatorname{tr}\left(\mathbf{a}\mathbb{V}\left[\vec{X}\right]\right) + \mathbb{E}\left[\vec{X}\right] \cdot \mathbf{a}\mathbb{E}\left[\vec{X}\right]$, for any random vector $\vec{X}$ and non-random square matrix $\mathbf{a}$.

optimum ψ^*, so it's not that useful as an estimate, but we can fix that by doing a parallel expansion of $M_n(\psi^*)$ around $\hat{\psi}_n$:

$$M_n(\psi^*) \approx M_n(\hat{\psi}_n) + \frac{1}{2}\left(\psi^* - \hat{\psi}_n\right) \cdot \mathbf{h}(\hat{\psi}_n)\left(\psi^* - \hat{\psi}_n\right) \tag{D.87}$$

$$\approx M_n(\hat{\psi}_n) + \frac{1}{2}\left(\psi^* - \hat{\psi}_n\right) \cdot \mathbf{h}(\psi^*)\left(\psi^* - \hat{\psi}_n\right) \tag{D.88}$$

$$\mathbb{E}\left[M_n(\psi^*)\right] \approx \mathbb{E}\left[M_n(\hat{\psi}_n)\right] + \frac{1}{2}\operatorname{tr}\left(\mathbf{h}(\psi^*)\mathbb{V}\left[\psi^* - \hat{\psi}_n\right]\right) \tag{D.89}$$

$$= \mathbb{E}\left[M_n(\hat{\psi}_n)\right] + \frac{1}{2}\operatorname{tr}\left(\mathbf{h}(\psi^*)\mathbb{V}\left[\hat{\psi}_n\right]\right) \tag{D.90}$$

$$= \mathbb{E}\left[M_n(\hat{\psi}_n)\right] + \frac{1}{2n}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.91}$$

$$m(\psi^*) \approx \mathbb{E}\left[M_n(\hat{\psi}_n)\right] + \frac{1}{2n}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.92}$$

Now we combine this with Eq. D.86:

$$\mathbb{E}\left[m(\hat{\psi}_n)\right] - \frac{1}{2n}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \approx \mathbb{E}\left[M_n(\hat{\psi}_n)\right] + \frac{1}{2}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.93}$$

$$\mathbb{E}\left[m(\hat{\psi}_n)\right] \approx \mathbb{E}\left[M_n(\hat{\psi}_n)\right] + \frac{1}{n}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.94}$$

Since $M_n(\hat{\psi}_n)$ is something we can observe, and its average equals $\mathbb{E}\left[M_n(\hat{\psi}_n)\right]$, we get

$$M_n(\hat{\psi}_n) + \frac{1}{n}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right) \tag{D.95}$$

as an unbiased estimator of $\mathbb{E}\left[m(\hat{\psi}_n)\right]$, and $n^{-1}\operatorname{tr}\left(\mathbf{j}\mathbf{h}^{-1}(\psi^*)\right)$ as our estimate of the optimism.

D.5.5 Application to Maximum Likelihood

All of the theory above applies straightforwardly to maximum likelihood estimation. As indicated in §§D.5.1.1 and D.5.2.1 above, $L_n(\theta)$ is the negative normalized log-likelihood for a parametric model with parameter θ, and $\hat{\theta}_n$ is the maximum likelihood estimate, and the limiting objective function $\ell(\theta)$ is the expected negative log-likelihood per observation, known in information theory [[CROSS-REF]] as the cross-entropy rate[11] The gradient $\nabla L_n(\theta)$ is called the **score** vector, often written U_n. The expected value of the Hessian is the **Fisher information**[12],

$$\mathbf{f}_n(\theta) \equiv \mathbb{E}\left[\nabla\nabla L_n(\theta)\right] \tag{D.96}$$

[11] The convergence $L_n(\theta) \to \ell(\theta)$ is called the "asymptotic equipartition" or "Shannon-McMillan-Breiman" property in information theory.

[12] It's more usual to write $I(\theta)$ or $\mathbf{I}(\theta)$, but I have found that this leads to confusion with the identity matrix $\mathbf{I}$, so I am using a slightly non-standard letter, while insisting that (i) matrices stay in bold type, and (ii) random variables are upper case and non-random ones lower-case.

Or, passing to the limit,

$$\mathbf{f}(\theta) = \nabla\nabla\ell(\theta) \tag{D.97}$$

With non-IID observations, many authors will use Eq. D.97 as the definition of $\mathbf{f}(\theta)$, which thus takes the role of $\mathbf{h}$ in our general estimation theory[13].

D.5.5.1 Consistency of the MLE

Let us focus a bit more on $\mathbb{E}[L_n(\theta)]$. The expectation here is taken not under the model's pdf[14] $f(\cdot;\theta)$ but the true, data-generating pdf $P(\cdot)$, so let's manipulate this a little.

$$\mathbb{E}[L_n(\theta)] = -\frac{1}{n}\int p(x_{1:n}) \log f(x_{1:n};\theta) dx_{1:n} \tag{D.98}$$

$$= -\frac{1}{n}\int p(x_{1:n}) \left[\log f(x_{1:n};\theta) - \log p(x_{1:n}) + \log p(x_{1:n})\right] dx_{1:n} \tag{D.99}$$

$$= \frac{1}{n}\left[\int p(x_{1:n}) \log \frac{p(x_{1:n})}{f(x_{1:n};\theta)} dx_{1:n} - \int TrueDensity(x_{1:n}) \log p(x_{1:n}) dx_{1:n}\right] \tag{D.100}$$

The second term in the square brackets does not depend on the model parameter θ, but only on the true, data-generating distribution; it is called the **entropy** of the distribution, often[15] written $H[X_{1:n}]$. The first term, which involves the average log probability ratio, is called the **relative entropy** between the distributions, or the **Kullback-Leibler** divergence of $f(\cdot;\theta)$ from $p(\cdot)$, written $D(p_n \| f_n(\theta))$. In symbols,

$$\mathbb{E}[L_n(\theta)] = \frac{1}{n}\left(H[X_{1:n}] + D(p_n \| f_n(\theta))\right) \tag{D.101}$$

Using facts about averages of logarithms, it's not too hard to show that $D(P\|Q) \geq 0$ for *any* two distributions P and Q, with $D(P\|Q) = 0$ if and only if $P = Q$. (See [[CROSS-REF TO EM]].) Here, this general fact about the divergence has the following implications:

1. $H[X_{1:n}]/n$ sets a lower limit on $\mathbb{E}[L_n(\theta)]$, regardless of the model class. If we measure how well our model is doing by (negative, normalized) log-likelihood, we can never do better than the (normalized) entropy.
2. Passing to the limit as $n \to \infty$, it's generally the case[16] that $n^{-1}H[X_{1:n}] \to h$, the **entropy rate**, and $n^{-1}D(p_n\|f_n(\theta)) \to d(\theta)$, the **relative entropy rate**

[13] Because Eq. D.97 involves normalizing by the number of observations, many writers call it the Fisher information *rate*, reserving "Fisher information" for $\mathbb{E}[-\nabla\nabla\Lambda_n]$. (Notice the minus sign which arises from using the log-likelihood, rather than the negative log-likelihood.) Call this last quantity $\mathcal{F}_n$. Then our $\mathbf{f} = \lim_{n\to\infty} n^{-1}\mathcal{F}_n$, and it is $\mathbf{f}$ which matters for asymptotics. (Of course for rigor one needs to show that the limit exists; if the data are IID, then $\mathcal{F}_n = n\mathbf{f}$ and the existence of the limit is immediate.)

[14] Or PMF if the data are discrete.

[15] This H should not be confused with the H of the Hessian.

[16] For instance, if the data-generating process is stationary.

or **divergence rate**[17]. This means that

$$\ell(\theta) = h + d(\theta) \tag{D.102}$$

where $d(\theta) \geq 0$.

3. If the model is well-specified, there is a unique special value of θ, call it θ^*, at which $f(\cdot;\theta^*) = p(\cdot)$. The expected (negative, normalized) log-likelihood is therefore optimized at the true parameter value, and only at the true parameter value. Similarly, in the limit, $\ell(\theta^*) = h$, and for every $\theta \neq \theta^*$, $\ell(\theta) > h$.

Combining the last item with our general theory of consistency (§D.5.3) tells us that when the model is well-specified, the MLE will generally be consistent, since the noisy objective function is converging to a limiting objective function whose unique optimum is the truth:

$$\hat{\theta}_n \to \theta^* \tag{D.103}$$

D.5.5.2 Mis-Specified Models and the Pseudo-Truth

If the model is *mis*-specified, the behavior of the MLE depends on the whether divergence rate function $d(\theta)$ has a unique minimum or not.

If $d(\theta)$ has a unique minimum, which I will keep writing θ^*, then our arguments from §D.5.3 still apply, and still lead to Eq. D.103. Only the interpretation changes. Instead of the MLE consistently estimating the truth, the MLE converges to the point in parameter space that minimizes the divergence. White (1994) gives this divergence-minimizing point the wonderful name of the **pseudo-true parameter value** or simply the **pseudo-truth**.

If, on the other hand, $d(\theta)$ has multiple minima, then the asymptotic behavior of the MLE will be more complicated. In general, it will wander between the minima. It may spend longer and longer amounts of time trapped in the vicinity of one minima before switching to another, and it may even get trapped around one minimum even if there are others (Berk, 1966). Again, all of this weirdness requires multiple global minima for $d(\theta)$, not just local minima.

We have, of course, been assuming that the limiting objective function has a *unique* minimum. Whether having multiple values of θ which all minimize the divergence from the true distribution is more or less plausible than having a correctly specified model is not for me to say.

D.5.5.3 Asymptotic Variance for the MLE

Assuming a unique smooth interior minimum to L_n and ℓ, we can use the general theory of §D.5.4 to get the asymptotic variance of the MLE. In general, this will be a sandwich covariance matrix, involving both the Hessian and the variance of the gradient.

[17] If the data are IID, then $H[X_{1:n}] = nh$, and similar for the divergence rate.

Fisher's Identity for Well-Specified Models

If the parametric model is correctly specified, then you can prove (Exercise D.1) **Fisher's identity**,

$$\mathbb{E}\left[\nabla\nabla L_n(\theta)\right] = \mathbb{V}\left[\nabla L_n(\theta)\right] \tag{D.104}$$

or, in terms of the symbols used above,

$$\mathbf{f}(\theta) = \mathbf{h}(\theta) = \mathbf{j} \tag{D.105}$$

Because this involves the Fisher information matrix, Eq. D.104 is also called the **information (matrix) identity**.

When Fisher's identity holds, the sandwich variance matrix simplifies from $\mathbf{h}^{-1}\mathbf{j}\mathbf{h}^{-1}$ to just $\mathbf{h}^{-1}(\theta^*) = \mathbf{j}^{-1} = \mathbf{f}^{-1}(\theta^*)$. Plugging in $\hat{\theta}_n \approx \theta^*$, one gets three approximations for the variance of the MLE:

$$\mathbb{V}\left[\hat{\theta}_n\right] \approx \mathbf{f}^{-1}(\hat{\theta}_n) \tag{D.106}$$

$$\approx \mathbf{H}_n^{-1}(\hat{\theta}_n) \tag{D.107}$$

$$\approx \mathbf{j}_n^{-1} \tag{D.108}$$

The first of these relies on being able to calculate the Fisher information matrix. The second approximates the Fisher information matrix by the the Hessian matrix of the (normalized, negative) log-likelihood at the MLE — what's called the **observed information matrix**. The third, rare form approximates it with the covariance matrix of the score vector[18]. Note, however, that this simplification relies on Fisher's identity holding, and it generally does *not* hold unless the model is well-specified.

This final thought suggest the possibility of *testing* whether the model is well-specified, by testing whether

$$\mathbf{H}_n^{-1}(\hat{\theta}_n) \approx \mathbb{V}\left[\nabla L_n(\hat{\theta}_n)\right] \tag{D.109}$$

Since this is a matrix equation, we need to think carefully about how to measure discrepancy between the two sides, and about what distribution it should have under the null hypothesis of correct specification — White (1994) is a good source for such details.

Efficiency and the Cramer-Rao Bound for Well-Specified Models

We've seen that when the parametric model is well-specified, the MLE is generally consistent (§D.5.5.1), and we've seen how to find its variance, using Fisher's identity to simplify the sandwich covariance. Since there could be other consistent estimators, it would be nice to know which consistent estimator has the smallest variance. [[FINISH]]

[18] See above, pp. 651, on ways to calculate $\mathbf{j}$ from data.

D.5.5.4 Asymptotic Distribution of the MLE

Provided the score vector ∇L_n is asymptotically Gaussian, the argument of §D.5.4.2 applies to the MLE. As indicated in that section, we can usually expect this to be the case when the data are IID (so L_n itself is a sample mean obeying the CLT) or only weakly dependent (ditto). It is worth noting, however, that for well-specified models, *all* that is required to get an asymptotically Gaussian MLE is for the log-likeliood to be asymptotically quadratic around the true parameter value[19] In any event, we then have Eq. D.81 for the asymptotic distribution of the MLE. If the model is well-specified, we can further simplify by using any of the approximations Eq. D.106–D.108 in place of the sandwich variance matrix. In short, for a well-specified, well-behaved model,

$$\hat{\theta}_n \rightsquigarrow \mathcal{N}(\theta^*, n^{-1}\mathbf{f}^{-1}(\hat{\theta}_n)) \tag{D.110}$$

This gives a test of the hypothesis that $\theta^* = \theta_0$ for any particular θ_0, and lets us form confidence sets around the MLE $\hat{\theta}_n$.

Notice, however, that nothing guarantees that the convergence to this asymptotic Gaussian distribution is especially fast — it may still be better to bootstrap.

D.5.5.5 Akaike's Information Criterion

Our treatment of the optimism (§D.5.4.3) applies to maximum likelihood estimators. The general Eq. D.95 says, in this case, that

$$L_n(\hat{\psi}_n) + \frac{1}{n}\,\mathrm{tr}\,(\mathbf{j}\mathbf{f}^{-1}(\hat{\theta}_n)) \tag{D.111}$$

is an unbiased estimate of the expected log-likelihood per observation. This can be compared across models, and used as a model-selection criterion. When doing so, it is somewhat more common to (in our terms) multiply through by n, yielding

$$nL_n(\hat{\psi}_n) + \mathrm{tr}\,(\mathbf{j}\mathbf{f}^{-1}(\hat{\theta}_n)) \tag{D.112}$$

as an unbiased estimate of the expected negative log-likelihood. Multiplying through by -1, and remembering that the log-likelihood is Λ_n,

$$\Lambda_n(\hat{\theta}_n) - \mathrm{tr}\,(\mathbf{j}\mathbf{f}^{-1}(\hat{\theta}_n)) \tag{D.113}$$

the log-likelihood penalized by the optimism, is an unbiased estimate of the expected log-likelihood. This is, in a model-selection context, often called **Takeuchi's information criterion** (TIC). (Whether one prefers a model that minimizes Eq. D.111 or D.112, or maximizes Eq. D.113, makes no difference.)

It's not exactly obvious, but it can be shown (Claeskens and Hjort, 2008, §2.9) that, for large n, TIC is asymptotically equivalent to using leave-one-out cross-validation to estimate the generalization error. For complicated models, Eq. D.111 can be much faster to calculate than leave-one-out, even considering the effort of getting $\mathbf{j}$ and $\mathbf{f}^{-1}$.

[19] This simple-sounding result rests on some deep work by Lucien Le Cam, accessibly presented in Geyer (2013). But even a crude vulgarization of Geyer's simplification of Le Cam would take too much space here.

As usual, we can simplify more if the model is well-specified. In that case, as we've just seen (§D.5.5.3), Eq. D.105 says $\mathbf{f}(\theta) = \mathbf{j}$, so $\mathrm{tr}\left(\mathbf{j}\mathbf{f}^{-1}(\psi^*)\right) = \dim(\psi)$, the number of parameters estimated. This gives the **Akaike information criterion**,

$$\Lambda_n(\hat{\psi}_n) - \dim(\psi) \tag{D.114}$$

where we just penalize the log-likelihood by the number of parameters[20].

AIC one was one the first tools proposed for systematic model selection, and has retained some truly devoted followers, especially among non-statisticians. I have mostly ignored it in this book, however, for specific reasons. As we've just seen, when the models are mis-specified, it's inferior to the TIC[21]. The TIC in turn is best thought of as a fast asymptotic approximation to leave-one-out cross-validation. Unfortunately, leave-one-out is known to be *in*-consistent for model selection — it over-fits, even as $n \to \infty$, and consequently so does AIC (Claeskens and Hjort, 2008, ch. 2). Against this, when all the models being considered are mis-specified, leave-one-out CV does select models with near-optimal predictive performance, and so TIC has virtues as a fast approximation to leave-one-out. But "TIC predicts well when models are all mis-specified, so we should use AIC, which simplifies TIC when the models are well-specified" is a twist of logic I just cannot follow.

D.5.5.6 Why Is the MLE Special?

Maximum likelihood estimation doesn't always work, but it does hold a special place in statistical theory and practice, for three reasons which should now be clear.

1. It gives us a recipe for building estimators from probability models.
2. It works a lot of the time.
3. When it works, it often works at least as well, statistically, as anything else.

Point (1), about the recipe, may sound trivial but it's actually important. If we're exploring some new area of data analysis, where we find ourselves needing to build new and/or complicated probability models, it helps a lot to have a systematic way to also build *an* estimator for the model, and especially an estimator where we know how to do inference. Otherwise we'd be left floundering, i.e., needing to think much harder.

Point (2) is what we've said about consistency (§D.5.5.1). The assumptions aren't trivial and they can fail, there are definitely times when the MLE is *not* consistent but other estimators are; yet all the same, the assumptions *do* hold a lot of the time.

Point (3) is what we've said about efficiency [[CROSS-REF]]. Knowing that

[20] In his original papers, Akaike wrote the equivalent of Eq. D.114 with a factor of 2 throughout. This is because standard asymptotics for likelihood ratio tests tell us that $2\times$ the log likelihood ratio should have a χ^2 distribution, and he wanted (differences in) his criterion to be comparable to this. This only becomes relevant when comparing numerical values of AIC from different sources, e.g., different pieces of software — you need to check whether the factor of 2 is there or not!

[21] Sometimes therefore called the **robust AIC**.

our estimator is giving us the smallest standard errors, or confidence sets, is telling us that we're using the information in the data well, which is intellectually comforting and often practically useful. Of course the assumptions needed for efficiency are even stronger than those for consistency. I should also add that I was careful to qualify the efficiency as *statistical*. Computing and then maximizing the likelihood can itself be a challenging computational task, and if the costs of computation are high enough, it may be than something which is less statistically efficient, but easier to compute, is better over-all. But this is a question of data-analytic practice which statistical theory has only just started to grapple with (Jordan, 2013).

D.6 Further Reading

My usual reference on optimization methods is Lange (2013), but there are many other good ones. Boyd and Vandenberghe (2004) is, deservedly, the standard reference on theory and algorithms for optimizing convex functions.

Spufford (2010) is the best historical novel about attempting to achieve utopia through the power of optimization and computers, and should be read by anyone trying to use data to change the world.

The material on optimizing noisy functions, and in particular on the behavior of maximum likelihood, is a simplified view of classical parts of theoretical statistics, much of it dating back to Fisher (1922); Cramér (1945) gives a detailed treatment, with references to the earlier literature. Most presentations require more advanced math (measure theory) than I use here; Barndorff-Nielsen and Cox (1995) and van der Vaart (1998) can be recommended as thorough, modern accounts. For the importance of considering what happens when the model is false, see Huber (1967) and White (1994). The simple approach to consistency of §D.5.3 comes from van der Vaart (1998, §5.2). The best single paper to read on this whole subject is Geyer (2013).

AIC was introduced by Akaike (1973); it was the first of many subsequent "information criteria". The best available over-view of model selection from a statistical perspective, covering the strengths, weaknesses, and inter-relations of (almost) all the information criteria, is Claeskens and Hjort (2008).

All of the arguments about estimation which involved Taylor series assumed, more or less implicitly, that the objective functions M_n and m had their minima in the *interior* of the parameter space. Matters become trickier, and rigor more necessary, when the optimum lives on the boundary of the parameter space — see, for instance, Self and Liang (1987).

Exercises

D.1 Let $p(x_{1:n};\theta)$ be a family of probability densities parameterized by a vector θ. Define $M_n(\theta) = -n^{-1}\log p(X_{1:n};\theta)$. Suppose that $X_{1:n}$ really are generated from this distribution, with parameter θ^*. Prove **Fisher's identity**:

$$\mathbb{E}\left[\nabla\nabla M_n(\theta^*)\right] = \mathbb{V}\left[\nabla M_n(\theta^*)\right] \tag{D.115}$$

taking all derivatives with respect to θ. Assume you can interchange differentiation and integration whenever it is convenient to do so. *Hints:* Integration by parts; all probability densities integrate to 1; $\partial \log f(x)/\partial x = \frac{1}{f(x)} \partial f/\partial x$.

Appendix E

Information Theory

Information theory is a branch of probability which studies the possibilities, and limits, of using random variables to convey information about other random variables or states of affairs — the use of random variables as *signals*. Many of its results, however, turn out to have implications in areas which at first sight have nothing to do with signals, and in particular with deep issues about uncertainty and inference. Because these ideas appear repeatedly when we look at issues like classification and density estimation, this appendix collects the most important, and basic, facts about information theory.

E.1 Entropy

E.1.1 Twenty Questions

You are probably familiar with the game of "twenty questions", where I think of an object, and you try to guess it by asking me questions, with your goal being to guess it in as few questions as possible. Information theory begins with a formalization of this children's game. My guess is drawn from a discrete set $\mathcal{X}$, with $|\mathcal{X}|$ distinct values, according to a probability distribution $p(x)$, so it's a random variable X. Your questions have to be binary, yes-or-no. What's the best you can do?

Since X is random, we're going to have to be content to do well in some average-or-typical sense. Let's agree that we want to minimize the *expected* number of binary questions needed to determine the value of X. Any *strategy*[1] for playing this game can be represented as a binary tree, where the interior nodes of the tree are questions, and the leaves are labeled by particular possible values x (Figure [[REF]]). If we write $L(x)$ for the length of the path to x, we want to find the tree which minimizes $\mathbb{E}[L(X)]$.

A little thought says that you can always pin down X by asking, at most, $\lceil \log_2 |\mathcal{X}| \rceil$ binary questions, since that many questions will pick out $2^{\lceil \log_2 |\mathcal{X}| \rceil} \geq |\mathcal{X}|$ values. But you could do better if you use the probability mass function $p(x)$. If $p(0) = 0.99$, for instance, it makes a lot of sense to ask first "Is $X = 0$?", regardless of the rest of the distribution. This might lead to asking some extra questions if $X \neq 0$, but we'd still come out ahead on average.

[1] Here, the alternative to a strategy is inconsistent improvisation, and, well, who knows how well that can do?

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

Some work (Exercise 1) shows that there's a lower bound to the expected number of binary questions in terms of the log probabilities: for any tree,

$$\mathbb{E}\left[L(X)\right] \geq -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{E.1}$$

This leads us to *define* a new quantity, called the **entropy**[2] or **self-information** of X:

$$H[X] \equiv -\sum_{x \in \mathcal{X}} p(x) \log_2 p(x) \tag{E.2}$$

with the understanding that $0 \log_2 0 = 0$ (see Exercise 3). The units are **bits** ("*b*inary dig*its*"); if you wanted to use natural logs, you'd just multiply all the values by a constant (what?), and call the units **nats**. You might wonder whether $H[X]$ is really a tight lower bound on the number of questions. Some more work (Exercise 2) shows that there is, in fact, always a coding tree where

$$\mathbb{E}\left[L(X)\right] \leq 1 + H[X] \tag{E.3}$$

E.1.2 Properties of Entropy

There are a number of easy-to-prove[3] properties of the entropy.

1. If the distribution is uniform, i.e., if $p(x) = \frac{1}{|\mathcal{X}|}$ for all x, then $H[X] = \log_2 |\mathcal{X}|$.
2. No matter what the distribution, $H[X] \leq \log_2 |\mathcal{X}|$, and if the distribution isn't uniform, the inequality is strict, $H[X] < \log_2 |\mathcal{X}|$ (Exercise 4).
3. If the distribution of X is degenerate, $p(x) = 1$ for some x, then $H[X] = 0$.
4. $H[X] \geq 0$, and, if the distribution is not degenerate, the inequality is strict, $H[X] > 0$ (Exercise **??**).
5. If $Y = f(X)$ for some one-to-one, invertible function f, then $H[X] = H[Y]$.
6. If $Y = f(X)$ for a many-to-one (and hence non-invertible) function f, then $H[X] > H[Y]$.
7.

Putting these together, we see that the entropy is a very natural way of measuring the *uncertainty* or *variability* of a (discrete) random variable. It's 0 when there's no uncertainty or variability; it's maximized when the distribution is uniform; it doesn't care how we label the discrete values, but it does decrease when we apply a many-one function.[4] Note, however, that this sense of the words "uncertainty" or "variability" is no more, and no less, subjective than it is when

[2] The name was borrowed from a quantity in physics, which has the same mathematical form; whether there is a substantial connection is a deep question. (See further reading.)

[3] I've made proving one of these a formal exercise, but you're invited to try your hand at all of them.

[4] A natural mathematical question is whether those good properties *force* us to use the entropy to measure the uncertainty of discrete variables. Khinchin (1957) provided a list of axioms for such measures, and showed that the entropy was, in fact, the only measure satisfying them all. One of the necessary axioms (having to do with the uncertainty of independent random variables) was a bit un-natural, and Rényi (1961) showed that weakening it leads to a whole family of measures, the "Rényi entropies". For more on this, see Aczél and Daróczy (1975).

we say things like "variance is a measure of how uncertain a random numerical variable is"[5].

E.1.3 Entropy, Expected Log Likelihood, and Cross Entropy

You might well still be wondering what this stuff about a guessing game is doing in a book about statistics. The big reason is that entropy turns out to have important statistical properties, which we'll see as we go along. As a sort of preview of coming attractions, however, think about the log-likelihood, which you remember is what we use to estimate (parametric) distributions efficiently.

Suppose our candidate probability mass function is $p(x)$, so the log-probability[6] function is $\log p(x)$. Since our data is random, X, the log-probability of what we actually observe will also be random, $\log p(X)$. As a random quantity, this will have some expected value. If p really is the true distribution, this expected value is just

$$\mathbb{E}\left[\log p(X)\right] = \sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{E.4}$$

If we've got independent samples $X_1, X_2, \ldots X_n$, then the law of large numbers tells us that

$$\frac{1}{n} \sum_{i=1}^{n} \log p(X_i) \rightarrow \mathbb{E}\left[\log p(X)\right] \tag{E.5}$$

Comparing Eq. E.4 to Eq. **??** tells us that

$$\mathbb{E}\left[\log p(X)\right] = -\frac{1}{\log 2} H[X] \tag{E.6}$$

As a slogan: "entropy is expected log probability". A consequence is of course that

$$-\frac{1}{n} \sum_{i=1}^{n} \log_2 p(X_i) \rightarrow H[X] \tag{E.7}$$

because we're assuming the X_i are IID, so the law of large numbers holds.

Now, you might well wonder what would happen if our candidate distribution *isn't* right. Suppose the truth is $p(x)$ but we consider $q(x;\theta)$ for various possible θ. Then expected log-likelihood will be

$$\mathbb{E}\left[logq(X;\theta)\right] = \sum_{x \in \mathcal{X}} p(x) \log q(x;\theta) \tag{E.8}$$

This is sometimes called the (negative) **cross-entropy**. By parallel reasoning to

[5] It should also be said that there are some situations where interpreting entropy as uncertainty, in any sort of subjective or psychological sense, leads to very peculiar results (Seidenfeld, 1987).

[6] We can't quite call it a log-likelihood yet, since in this paragraph we're only considering a single probability distribution, and purists define likelihood as a function over *multiple* possible distributions.

the last paragraph,

$$\frac{1}{n} sum_{i=1}^{n} \log q(X_i; \theta) \rightarrow \mathbb{E}\left[logq(X; \theta)\right] \tag{E.9}$$

So, if maximum likelihood estimation is going to work, it had better be the case that the cross-entropy is uniquely minimized by the θ_0 where $q(x; \theta_0) = p(x)$. We will see later (§E.2.3) that this is, indeed, the case.

E.1.4 Multiple Variables: Joint and Conditional Entropy

The definition of the entropy extends naturally to multiple variables. The **joint entropy** of any collection of discrete random variables $X_1, X_2, \ldots X_k$ is Eq. **??** applied to the joint distribution:

$$H[X_1, X_2, \ldots X_k] = -\sum_{x_1 \in \mathcal{X}_1} \sum_{x_2 \in \mathcal{X}_2} \ldots \sum_{x_k \in \mathcal{X}_k} \Pr\left(X_1 = x_1, X_2 = x_2, \ldots X_k = X_k\right) \log_2 \Pr\left(X_1 = x_1, X_2 = \right. \tag{E.10}$$

In terms of the guessing game, this is the average number of questions we need to determine, *simultaneously*, the value of all of the variables. This tells us that the joint entropy is at most the sum of the marginal entropies:

$$H[X_1, X_2, \ldots X_k] \leq \sum_{i=1}^{k} H[X_i] \tag{E.11}$$

Equality here implies that the random variables are all independent of each other (Exercise 6).

We can also apply the definition of entropy to a conditional[7] distribution:

$$H[Y|X = x] = -\sum_{y \in \mathcal{Y}} \Pr\left(Y = y|X = x\right) \log_2 \Pr\left(Y = y|X = x\right) \tag{E.12}$$

This is the uncertainty or variability remaining in Y, given that X has taken the particular value x. What is usually called the **conditional entropy** is the expected value of this:

$$H[Y|X] = \sum_{x \in \mathcal{X}} \Pr\left(X = x\right) H[Y|X = x] \tag{E.13}$$

We think of the conditional entropy as the average uncertainty or variability remaining in Y, once X has been pinned down. You can check (Exercise 7) that

$$H[X, Y] = H[Y|X] + H[X] = H[X|Y] + H[Y] \tag{E.14}$$

and, consequently,

$$H[Y|X] \leq H[Y] \tag{E.15}$$

[7] I have written these formulas with just one conditioning variable for simplicity, but the same ideas apply when we condition on any number of random variables.

That is, conditioning reduces the (expected) uncertainty. One consequence of these definitions, and of the basic properties of entropy, is that

$$H[Y|X] = 0 \tag{E.16}$$

if, and only if, Y is a function of X.

E.1.5 Mutual Information

When we have two variables X and Y, we can ask by how much knowledge of X reduces the uncertainty in Y, on average. This is clearly

$$H[Y] - H[Y|X] \tag{E.17}$$

so we can think of this as the amount of *information* X carries about Y. By manipulating the definitions (Exercise **??**, you can check that

$$H[Y] - H[Y|X] = H[X] - H[X|Y] \tag{E.18}$$

Since the amount of information is symmetric, it's called the **mutual information**:

$$I[X;Y] = H[Y] - H[Y|X] = H[X] - H[X|Y] \tag{E.19}$$

Mutual information inherits some very important properties from the entropy:

1. $I[X;Y] \geq 0$.
2. $I[X;Y] = 0$ if and only if $X \perp\!\!\!\perp Y$.
3. For any function f, $I[f(X);Y] \leq I[X;Y]$.
4. For any $1-1$ function f, $I[f(X);Y] = I[X;Y]$ (but there can be many-one functions which preserve mutual information; Exercise **??**.)

Further manipulating the definitions (Exercise 9) shows that

$$I[X;Y] = \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} \Pr(X = X, Y = y) \log_2 \frac{\Pr(X = x, Y = y)}{\Pr(X = x)\Pr(Y = y)} \tag{E.20}$$

This tells us that the mutual information is an average log-likelihood ratio, between the actual joint distribution and the product of the marginal distributions. In this sense, the mutual information measures how far X and Y are from being independent. This equation also explains why the ratio

$$\log_2 \frac{\Pr(X = x, Y = y)}{\Pr(X = x)\Pr(Y = y)} \tag{E.21}$$

is sometimes called the **pointwise mutual information** at x and y.

E.1.5.1 Conditional Mutual Information

We can ask how much information X gives us about Y, *given* that we already know a third variable Z. This **conditional mutual information** is

$$I[X;Y|Z] = H[Y|Z] - H[Y|X,Z] \tag{E.22}$$

(You can check that this is equal to $H[X|Z] - H[X|Y,Z]$.) Just as mutual information measures how far X and Y are from being independent, conditional mutual information measures distance from *conditional* independence:

1. $I[X;Y|Z] \geq 0$
2. $I[X;Y|Z] = 0$ if and only if $X \perp\!\!\!\perp Y|Z$.

E.1.6 Entropy and Mutual Information for Continuous Variables

So far, we've defined everything for discrete random variables. The definitions however carry over naturally to continuous random variables, with probability density function p:

$$H[X] \equiv -\int p(x)\log_2 p(x)dx \tag{E.23}$$

$$H[X,Y] \; equiv -\int p(x,y)\log_2 p(x,y)dxdy \tag{E.24}$$

$$H[Y|X=x] \equiv -\int p(y|x)\log_2 p(y|x)dy \tag{E.25}$$

$$H[Y|X] \equiv \int p(x)H[Y|X=x]dx \tag{E.26}$$

$$I[X;Y] \equiv H[Y] - H[Y|X] \tag{E.27}$$

$$I[X;Y|Z] \equiv H[Y|Z] - H[Y|X,Z] \tag{E.28}$$

Entropy[8] still measures how spread out a distribution is, and so how much uncertainty there is in a random draw from that distribution. Conditional entropy still measures how much of this uncertainty remains after conditioning one variable on another. Mutual information still measures how much the entropy is reduced by conditioning. However, there are some important differences, which basically arise because the numerical values of probability densities change when we change our units of measurement, while probability mass functions do not.

1. $H[X]$ can be < 0; in particular, if the distribution puts probability 1 on a single point, $H[X] = -\infty$
2. If f is $1-1$, it is not necessarily the case that $H[X] = H[f(X)]$. (For instance, let $f(x) = 2x$, and calculate the change in entropy.)
3. If X has a uniform distribution over a region of size v, then $H[X] = \log_2 v$.

On the other hand, a lot of things work just the same.

[8] Some people insist on referring to the entropy of a continuous random variable as a **differential entropy**, and write it with a lower-case h.

1. $H[X, Y] = H[Y|X] + H[X]$.
2. $H[X, Y] \leq H[X]+H[Y]$, with equality if and only if X and Y are independent.
3. $H[Y|X] \leq H[Y]$, with equality if and only if X and Y are independent.
4. $I[X; Y]$ is symmetric.
5. $I[X; Y] \geq 0$, with $I[X; Y] = 0$ if and only if $X \perp\!\!\!\perp Y$.
6. For any function f, $I[f(X); Y] \leq I[X; Y]$.
7. For any $1-1$ function f, $I[f(X); Y] = I[X; Y]$, but there can be many-1 functions which also preserve mutual information.
8. $I[X; Y|Z] \geq 0$, with $I[X; Y|Z] = 0$ if and only if $X \perp\!\!\!\perp Y|Z$.

E.2 Kullback-Leibler Divergence

The entropy is a property of one probability distribution; the mutual information is a property of a *joint* distribution. What if we want to *compare* two distributions? One way to do it is to use what's called the **Kullback-Leibler (KL) divergence**. For two distributions P and Q, this is

$$D(P\|Q) \equiv \sum_x p(x) \log \frac{p(x)}{q(x)} \tag{E.29}$$

when the space is discrete, and p and q are the probability mass functions, and

$$D(P\|Q) \equiv \int_x p(x) \log \frac{p(x)}{q(x)} dx \tag{E.30}$$

when the space is continuous, and p and q are the probability densities.[9]

The divergence has a number of easily-checked properties.

1. The divergence is non-negative: $D(P\|Q) \geq 0$.
2. Zero divergence means the distributions are equal: $D(P\|Q) = 0$ if and only if $P = Q$.
3. The divergence is convex: $D(P\|\alpha Q_1+(1-\alpha)Q_2) \leq \alpha D(P\|Q_1)+(1-\alpha)D(P\|Q_2)$.
4. Divergence never grows under transformations: if $T = t(X)$, then

$$D_T(P\|Q) \leq D_X(P\|Q) \tag{E.31}$$

(This is sometimes called the "data processing inequality"). The divergence stays equal if t is a $1-1$ function. More generally, if t preserves the divergence, then T is a **sufficient statistic**.

[9] The similarity of these equations suggests that there should be some common form, and there is, but it needs advanced (measure-theoretic) probability to define. It goes as follows. Suppose P and Q are probability measures, and P "dominates" Q, i.e., $P(A) = 0$ implies $Q(A) = 0$ for all sets A. Then there exists a Radon-Nikodym derivative dQ/dP, i.e., a function such that $Q(A) = \int \frac{dQ}{dP}(x) dP(x)$. (In fact, there generally exist infinitely many such functions, differing on sets of P-measure 0.) We then define $D(P\|Q) \equiv -\int \log \frac{dQ}{dP}(x) dP(x)$. Finally, if P does not dominate Q — that is, if Q gives positive measure to set of P-measure 0 — then we set $D(P\|Q) = \infty$.

5. Divergence is additive across independent variables: if P and Q are joint distributions for $X_1, \ldots X_k$, and those variables are independent under both P and Q, then

$$D_{X_1,\ldots X_k}(P\|Q) = \sum_{i=1}^{k} D_{X_i}(P\|Q) \tag{E.32}$$

The last property brings us to the notion of a divergence *rate* between two stochastic processes. If $X_1, X_2, \ldots X_n, \ldots$ are IID under both P and Q, then

$$D_{X_1,\ldots X_n}(P\|Q) = nD_{X_1}(P\|Q) \tag{E.33}$$

and

$$\lim_{n\to\infty} \frac{1}{n} D_{X_1,\ldots X_n}(P\|Q) = D_{X_1}(P\|Q) \tag{E.34}$$

is the rate of divergence, which you can think of as the amount of information we get for discriminating between P and Q per observation. More generally, whenever

$$d(P\|Q) \equiv \lim_{n\to\infty} \frac{1}{n} D_{X_1,\ldots X_n}(P\|Q) \tag{E.35}$$

exists, we call it the divergence rate, and it has the same interpretation. A divergence rate generally exists between any two stationary processes, but stationarity is not necessary for the rate to exist[10].

E.2.1 Recovering Entropy from Divergence

We can define entropy in terms of divergence, at least for discrete variables. Set U to be the uniform distribution on $\mathcal{X}$, which is a set of size (say) k. Then

$$D(P\|U) = \sum_{x\in\mathcal{X}} p(x) \log \frac{p(x)}{1/k} \tag{E.36}$$

$$= \sum_{x\in\mathcal{X}} p(x) \log p(x) + \log k \sum_{x\in\mathcal{X}} p(x) \tag{E.37}$$

so

$$H[X] = \log k - D(P\|U) \tag{E.38}$$

E.2.2 Mutual Information and Divergence

If we have a joint distribution P over two variables X and Y, we can extract marginal distributions from it, P_X and P_Y. We can then look at the product of those marginal distributions, $P_X \otimes P_Y$. This last is the distribution with the same

[10] For a detailed treatment of this question, see Gray (1990).

marginals as P, but where X and Y are independent. If we look at the divergence of $P_X \otimes P_Y$ from P, it's

$$D(P\|P_X \otimes P_Y) = \sum p(x,y) \log_2 \frac{p(x,y)}{p(x)p(y)} \tag{E.39}$$

But this is just Eq. E.20 for the mutual information. So

$$I[X;Y] = D(P\|P_X \otimes P_Y) \tag{E.40}$$

(You might ask what $D(P_X \otimes P_Y\|P)$ is. It appears to have no simple formula in terms of entropies, but can be important in testing whether $X \perp\!\!\!\perp Y$. (Palomar and Verdú, 2008) call it the "lautum information".)

E.2.3 Divergence and Expected Log-Likelihood Ratios

Every probability model gives us a family of distributions for the data, say Q_θ where θ is the parameter (in a general sense; it might be infinite-dimensional). When we see the data x, we assign the specific parameter value θ the likelihood $q(x;\theta)$, and the negative log-likelihood $-\log q(x;\theta)$). Since X is random, it makes sense to ask what the expected value of the log-likelihood is. The answer will, obviously, depend on the true distribution of X. For the discrete case, this is

$$-\mathbb{E}_P\left[\log q(X;\theta)\right] = -\sum_{x\in\mathcal{X}} p(x) \log q(x;\theta) \tag{E.41}$$

$$= -\sum_{x\in\mathcal{X}} p(x) \log \left(\frac{q(x;\theta)}{p(x)} p(x)\right) \tag{E.42}$$

$$= -\sum_{x\in\mathcal{X}} p(x) \log \frac{q(x;\theta)}{p(x)} - \sum_{x\in\mathcal{X}} p(x) \log p(x) \tag{E.43}$$

$$= D(P\|Q_\theta) + H_P[X] \tag{E.44}$$

where the sub-scripts are reminders about which distribution is used to calculate expectations. The same result holds true for the continuous case.

In words, the expected negative log-likelihood equals the entropy plus the Kullback-Leibler divergence. The entropy sets a lower bound on the expected negative log-likelihood, because $D(P\|Q) \geq 0$. The divergence says how far the distribution Q is from meeting this lower bound. Minimizing KL divergence is therefore the same as maximizing *expected* log-likelihood, and vice versa.

Using the same sort of reasoning about coding that we went through for the entropy, you can convince yourself that $-\mathbb{E}_P\left[\log q(X;\theta)\right]$ is the expected length of encodings (= number of questions) when X is actually drawn from the distribution P, but we base our code (= questions) on the distribution Q_θ. This is what we saw in §E.1.3 as the cross-entropy.

With independent observations $X_1, X_2, \ldots X_n$, the law of large numbers leads us to believe that

$$-\frac{1}{n} \log q(X_i;\theta) \to H_P[X_1] + D(P\|Q_\theta) \tag{E.45}$$

If the model is well-specified, $P = Q_{\theta_0}$ for some true θ_0. In that case, the divergence has a unique minimum (=0), and the expected log-likelihood has a unique maximum (= $-H_P$), both located at θ_0. If the convergence in Eq. E.45 is well-behaved, and the divergence $D(P\|Q_\theta)$ is well-behaved as a function of θ, then maximum likelihood estimates have to converge on θ_0. (A more specific statement is a little bit more technical, so it goes in the next sub-section, which can be skipped without loss of continuity.)

E.2.3.1 Convergence of Maximum Likelihood[11]

Suppose we have observations $X_1, \ldots X_n$. For each θ in our family of models, we define the normalized negative log-likelihood as

$$L_n(\theta) \equiv -\frac{1}{n} \log q(X_i; \theta) \tag{E.46}$$

This is a random quantity, hence the capital letter. We've just seen that

$$\mathbb{E}\left[L_n(\theta)\right] = H_P[X_1] + D(P\|Q_\theta) \tag{E.47}$$

Abbreviate the right-hand side as $\lambda(\theta)$. The maximum likelihood estimate is

$$\hat{\theta}_n \equiv \operatorname{argmin} L_n(\theta) \tag{E.48}$$

The best value of the parameter, θ_0, is

$$\theta_0 \equiv \operatorname{argmin} \lambda(\theta) = \operatorname{argmin} D(P\|Q_\theta) \tag{E.49}$$

Here is a pair of conditions which are, together, enough to ensure that $\hat{\theta}_n \to \theta_0$.

1. $L_n(\theta)$ converges[12] **uniformly** to $\lambda(\theta)$. That is, $\sup_\theta |L_n(\theta) - \lambda(\theta)| \to 0$.
2. $\lambda(\theta)$ has a **well-separated minimum**: that is, have a value of the divergence close to the minimum implies being located close to the minimum. More formally, for all sufficiently small ϵ, there is a $\delta > 0$ such that $\lambda(\theta) - \lambda(\theta_0) < \epsilon$ implies $|\theta - \theta_0| < \delta$.

Then $\hat{\theta}_n \to \theta_0$. To prove this, we want to use the well-separated minimum property, so we want to show that, eventually, $\lambda(\hat{\theta}_n) - \lambda(\theta_0) < \epsilon$, for any ϵ. To do that, we do some algebra:

$$\lambda(\hat{\theta}_n) - \lambda(\theta_0) = \lambda(\hat{\theta}_n) - L_n(\hat{\theta}_n) + L_n(\hat{\theta}_n) - \lambda(\theta_0) \tag{E.50}$$

$$= \lambda(\hat{\theta}_n) - L_n(\hat{\theta}_n) + L_n(\hat{\theta}_n) - L_n(\theta_0) + L_n(\theta_0) - \lambda(\theta_0) \tag{E.51}$$

$$\leq \lambda(\hat{\theta}_n) - L_n(\hat{\theta}_n) + L_n(\theta_0) - \lambda(\theta_0) \tag{E.52}$$

because $L_n(\hat{\theta}_n) \leq L_n(\theta_0)$ by the definition of $\hat{\theta}_n$. In turn,

$$\lambda(\hat{\theta}_n) - L_n(\hat{\theta}_n) + L_n(\theta_0) - \lambda(\theta_0) \leq 2 \sup_\theta |L_n(\theta) - \lambda(\theta)| \tag{E.53}$$

[11] This is shamelessly ripped off from van der Vaart (1998).

[12] Purists at this point would want to ask about the mode of convergence — in probability, almost sure, L_p? They may amuse themselves by checking whether the argument works for their favorite convergence mode.

But we know that this supremum becomes arbitrary small, in particular smaller than $\epsilon/2$, by uniform convergence. Q.E.D.

It's worth noting that this argument doesn't require independent observations. It also doesn't really require that what we're minimizing is negative log-likelihood. Rather, the objective function L_n just has to converge to *some* limiting function λ which has a well-separated minimum, and that convergence has to be uniform in θ. In fact, we can weaken even that requirement. It would be enough if, with probability $\to 1$, $\hat{\theta}_n$ falls within some set G where convergence is uniform, and $\theta_0 \in G$.

E.2.4 Divergence and Hypothesis Testing

Suppose we want to test the null hypothesis that $X \sim P$ against the alternative that $X \sim Q$. The divergences $D(P\|Q)$ and $D(Q\|P)$ turn out to control the error rates of all possible hypothesis tests; the bigger the divergences, the lower the possible error rates.

Any test will either retain P or reject it, based on the value of X we observe. So we can think of the test as a function $T(X)$, which takes two values, $T(X) = 1$ when X falls into the rejection region, and $T(X) = 0$ otherwise. We know (Eq. E.31) that $D_X(P\|Q) \geq D_T(P\|Q)$, and likewise $D_X(Q\|P) \geq D_T(Q\|P)$, so

$$D_X(P\|Q) \geq P(T=0)\log\frac{P(T=0)}{Q(T=0)} + P(T=1)\log\frac{P(T=1)}{Q(T=1)} \quad \text{(E.54)}$$

$$D_X(Q\|P) \geq Q(T=0)\log\frac{Q(T=0)}{P(T=0)} + Q(T=1)\log\frac{Q(T=1)}{P(T=1)} \quad \text{(E.55)}$$

But the event $T = 1$ is rejecting the null, so $P(T = 1)$ is the probability of rejecting the null when it's true, i.e., of a false rejection or a Type I error, called the **size** of the test and usually written α. Likewise, $Q(T = 1)$ is the probability of rejecting the null when it's false, of correct rejection, or the **power** of the test, usually written β. So

$$D_X(P\|Q) \geq (1-\alpha)\log\frac{1-\alpha}{1-\beta} + \alpha\log\frac{\alpha}{\beta} \quad \text{(E.56)}$$

$$D_X(Q\|P) \geq (1-\beta)\log\frac{1-\beta}{1-\alpha} + \beta\log\frac{\beta}{\alpha} \quad \text{(E.57)}$$

To learn about the error rates from these inequalities, remember that the right-hand sides are divergences between distributions between binary variables. This means that they reach their minimum, of zero, when $\alpha = \beta$, are > 0 whenever $\alpha \neq \beta$, and increase as we move away from $\alpha = \beta$. In particular, holding α fixed, the right-hand side of Eq. E.56 is increasing in β, for $\beta > \alpha$, and goes to infinity as $\beta \to 1$. So for each value of α and $D(P\|Q)$, there is a maximum possible value of β. Similarly, for each value of β and $D_X(Q\|P)$, there is a minimum attainable value of α. (Consider what happens to the right-hand side of Eq. E.57 as $\alpha \to 0$.) In other words:

- $D_X(P\|Q)$ controls the maximum possible test power β^*.
- $D_X(Q\|P)$ controls the minimum possible test size α^*.

There is no easy way to invert Eqs. E.56 and E.57 to get β^* and α^*. However, if we have a growing number of observations, we can say a little about the asymptotics. If X is really $X_1, X_2, \ldots X_n$, then Eq. E.56 tells us that, for fixed α,

$$D_{X_1,X_2,\ldots X_n}(P\|Q) = \beta_n^* \log \frac{\beta_n^*}{\alpha} + (1-\beta_n^*) \log \frac{1-\beta_n^*}{1-\alpha} \tag{E.58}$$

Let's abbreviate $D_{X_1,X_2,\ldots X_n}$ as D_n. Clearly, if the divergence tends to a finite limit as n grows, $\lim_n D_n < \infty$, then that puts a limit on how high β^* can ever grow. Equally clearly, if $D_n \to \infty$, then $\beta_n^* \to 1$. In that case, we can even say something about the rate at which β_n^* approaches 1, or rather the rate at which $1-\beta_n^*$ approaches zero:

$$-\lim_{n\to\infty} \frac{1}{n} \log(1-\beta^* n) = \lim_n \to \infty \frac{1}{n} D_n(P\|Q) \tag{E.59}$$

(Exercise 10.) Notice that while α matters for the exact value of β_n^*, it drops out here, where we're just considering the exponential rate at which β_n^* approaches 1.

Parallel reasoning applies to α^*: it goes to zero (for fixed β) if and only if $D_n(Q\|P) \to \infty$, in which case

$$-\lim_{n\to\infty} \frac{1}{n} \log(\alpha_n^*) = \lim_n \to \infty \frac{1}{n} D_n(Q\|P) \tag{E.60}$$

If we have IID observations, these limits are of course just $D_1(P\|Q)$ and $D_1(Q\|P)$.

E.2.5 Divergence and Fisher Information

[[Notes taken from a different course; fix up for uniform notation]]

Suppose we have a family of probability distributions parameterized by a finite-dimensional manifold. Let's call the indexing parameter θ, with the corresponding probability measure P_θ, with corresponding probability densities[13] $p_\theta(x)$. We pick out one parameter value as our favorite, and without loss of generality call it 0. We would like to know how well we can discriminate between 0 and a slight perturbation, say θ. We would like to measure this discirmination in terms of the relative entropy, a.k.a. Kullback-Leibler divergence. Can we express the relative entropy in terms of θ, at least if θ is in some sense small?

In a slight abuse of notation, write $D(\theta)$ as an abbreviation for $D(P_0\|P_\theta)$. Taylor expand this to second order[14]:

$$D(\theta) = D(0) + \theta \cdot D'(0) + \frac{1}{2}\theta \cdot D''(0)\theta + O(\|\theta\|^3) \tag{E.61}$$

[13] If you know what it means, you should assume that all the P_θ are absolutely continuous with respect to a common σ-finite reference measure, and apply the Radon-Nikodym theorem. If you don't, just assume that the pdfs are well-behaved.

[14] Thereby assuming that all the relevant derivatives exist and are integrable, etc.

where $\cdot$ is inner product, and D' and D'' are the appropriate matrices of first and second partial derivatives. What are they?

Recall that

$$D(\theta) = \mathbf{E}_0\left[\log \frac{p_0(X)}{p_\theta(X)}\right] = \mathbf{E}_0\left[\log p_0(X)\right] - \mathbf{E}_0\left[\log p_\theta(X)\right]$$

where I'm assuming that nothing blows up. So when I take derivatives with respect to θ, the first term, which involves only p_0, goes away.

Then

$$\frac{\partial D(\theta)}{\partial \theta_i} = -\frac{\partial}{\partial \theta_i}\mathbf{E}_0\left[\log p_\theta(X)\right] \tag{E.62}$$

$$= -\frac{\partial}{\partial \theta_i}\int dx\ p_0(x)\log p_\theta(x) \tag{E.63}$$

$$= -\int dx\ p_0(x)\frac{\partial \log p_\theta(x)}{\partial \theta_i} \tag{E.64}$$

$$= -\int dx\ p_0(x)\frac{1}{p_\theta(x)}\frac{\partial p_\theta(x)}{\partial \theta_i} \tag{E.65}$$

(Eq. E.64 assumes enough regularity that we can "differentiate under the integral sign".) Evaluated at $\theta = 0$, we get

$$\int dx\ \left.\frac{\partial p_\theta(x)}{\partial \theta_i}\right|_{\theta=0}$$

which is clearly zero (since $p_\theta(x)$ integrates to 1 for all θ). So $D'(0) = 0$.

As for D'', assume we can interchange expectation and a second derivative:

$$\frac{\partial^2 D(\theta)}{\partial \theta_i \partial \theta_j} = -\frac{\partial^2}{\partial \theta_i \partial \theta_j}\mathbf{E}_0\left[\log p_\theta(X)\right] \tag{E.66}$$

$$= -\mathbf{E}_0\left[\frac{\partial^2 \log p_\theta(X)}{\partial \theta_i \partial \theta_j}\right] \tag{E.67}$$

$$\equiv F_{ij}(\theta) \tag{E.68}$$

where $F(\theta)$ is the Fisher information matrix at θ.

Putting everything together,

$$D(P_0\|P_\theta) \approx \frac{1}{2}\theta \cdot F(0)\theta \tag{E.69}$$

when θ is small. In words, the Fisher information says how much relative entropy is produced by a small perturbation to the parameters.

E.3 Convexity and Concavity

Many useful properties of entropy and divergence are most easily proved by using more general facts about convex and concave functions. These facts are also important parts of general mathematical culture, so they're worth going over in their own right.

A set C is **convex** when, for any two points x_1 and x_2 in C, and for real number $0 \leq a \leq 1$, $ax_1 + (1-a)x_2$ is also in C. (Note that this definition applies whenever we can make sense of multiplying objects by real numbers, and adding objects, i.e., whenever we're dealing with a vector space.) The point $ax_1 + (1-a)x_2$ is called a **convex combination** of x_1 and x_2. This implies that convex combinations of any number of points in C remain in C (Exercise 11).

A one-dimensional[15] function $f(x)$ is called "convex" when the region above the curve is a convex set. This is equivalent to a number of other properties:

1. The line between the points $(x_1, f(x_1))$ and $(x_2, f(x_2))$ remains above, or on, the curve of $f(x)$ between those two points.
2. "The value of the function at a convex combination is less than or equal to the convex combination of the function values": When $0 \leq a \leq 1$,

$$f(ax_1 + (1-a)x_2) \leq af(x_1) + (1-a)f(x_2) \tag{E.70}$$

   If, whenever $0 < a < 1$, the inequality is strict, the function is called **strictly convex**.
3. For any pair x_1, there is a number s such that, for any x_2,

$$f(x_2) \geq f(x_1) + s(x_2 - x_1) \tag{E.71}$$

   (Exercise 12))
4. If f is differentiable, with derivative f', then for any x_1 and any x_2,

$$f(x_2) \geq f(x_1) + f'(x_1)(x_2 - x_1) \tag{E.72}$$

   i.e., the function lies above what would be suggested by a first-order Taylor expansion (Exercise 12; for Taylor expansions, see Appendix B).
5. If f has two derivatives, then $f''(x) \geq 0$ for all x. (Note that there are important examples, like step functions, which don't have second derivatives [everywhere], so convexity isn't just an obfuscated way of talking about having a positive second derivative.)
6. If f and g are both convex functions, and $a, b \geq 0$, then

$$af(x) + bg(x) \tag{E.73}$$

   is also convex.
7. If f is convex and g is non-decreasing, then $g(f(x))$ is convex.

If $-f$ is convex, then f is **concave**.

All convex functions share a number of important properties.

1. Any local minimum of a convex function, on a convex set, is also a global minimum on that set (Exercise 151).
   1. Conversely, any local maximum of a concave function over a convex set is also a global maximum over that set.

[15] All of this carries over naturally to functions of more than one argument, but at more cost in notation and/or verbiage than is useful here.

   2. A strictly convex function has (at most) one global minimum over each convex set (Exercise 154).
2. The **sublevel sets** of f are the regions where $f(x) \leq a$, or $f(x) < a$. If f is convex, these are convex sets (Exercise 14). (The converse does not follow, but that's harder to show.)
3. **Jensen's inequality**: if f is convex, then "the expected value of the function is bigger than the function at the expected value":

$$\mathbb{E}[f(X)] \geq f(\mathbb{E}[X]) \tag{E.74}$$

   no matter what the distribution of X. Conversely, if f is concave, then

$$\mathbb{E}[f(X)] \leq f(\mathbb{E}[X]) \tag{E.75}$$

Actually finding the minimum of a convex function over a convex set is the computational problem of **convex programming**; efficient algorithms exist, but are beyond what we need here.

E.4 Further Reading

The best, and also standard, textbook on information theory is Cover and Thomas (2006); my notation follows theirs. This book also covers many interesting and important topics I have deliberately omitted for reasons of space, such as how to actually design efficient codes, error-correction, and algorithmic (non-probabilistic) information theory.

Information theory, in the modern sense, emerged almost fully formed in one foundational paper, Shannon (1948). This was, itself, collecting previously-classified work done during World War II. There were, naturally, earlier efforts which (at least in retrospect) seem like anticipations, such as Hartley (1928), but Shannon's influence was immediate and decisive.

The main conceptual development post-Shannon came very quickly: it was Kullback and Leibler's notion of "divergence", introduced in Kullback and Leibler (1951), and whose statistical implications were thoroughly explored in Kullback (1968). (Unlike Cover and Thomas, or Shannon's original paper, these works make extensive use of advanced, measure-theoretic probability.)

Entropy has a much longer history in physics, where it has at least three different definitions[16], one of which[17] just is the Shannon entropy of a certain distribution. For an outstanding introduction to entropy in physics, and to a lot of related, fascinating science, see Sethna (2006). The late E. T. Jaynes very forcefully advocated the position that the equality of Shannon entropy with (one sort of) physical entropy was deeply important, going so far as to found a whole approach to statistics on the basis of a "principle of maximum

[16] The thermodynamic entropy of Clausius, defined in terms of temperature and heat flow; the Boltzmann entropy, defined in terms of the volume or number of microscopic molecular states compatible with a given macroscopic state; and the Gibbs entropy, defined in terms of distributions over microscopic states.

[17] That of Gibbs.

entropy" (Jaynes, 1983, 2003). This school has many more advocates among physicists than among statisticians, for reasons best explored elsewhere (`http://bactra.org/notebooks/max-ent.html`).

On convexity, the classic reference is Rockafellar (1970). For statistical purposes, most of what's relevant in convex analysis can be better learned from Boyd and Vandenberghe (2004), which also discusses the theory and practice of optimizing convex functions. Convexity, and convex optimization, also plays a very important role in economics; Exercise 17 offers a sense of why. In fact, the importance of convex optimization to economics inspired the *last* period when a huge number of very bright, very idealistic people armed with data, computers and algorithms thought they were going to optimize their way to utopia, as dramatized by Spufford (2010), a book I earnestly recommend to all readers of this one.

E.5 Exercises

1. *Twenty questions and entropy* As in §E.1.1, we consider the game of figuring out the value of a discrete random variable X, taking values in a set $\mathcal{X}$ of size (cardinality) k, by asking binary questions. Any strategy for playing this can be represented by a binary tree, where the internal nodes represent questions, and the leaves represent possible values x; each internal node has two child nodes. Fix on your favorite tree, and write $L(x)$ for the length of the path to leaf-node x.
    1. Show that there is always a tree where the maximum value of $L(x)$ is $\lceil \log_2 k \rceil$, i.e., $\log_2 k$ rounded up to the nearest whole number. This means showing that there is always a tree where $\log_2 k \leq \max_x L(x) \leq \log_2 k + 1$. Then use this to show that we can always achieve $\mathbb{E}\left[L(X)\right] \leq \lceil \log_2 k \rceil$. *Hint:* First show that when k is exactly a power of 2, there is always a tree where $L(x) = \log_2 k$ for all x.
    2. Show that for any tree, $\sum_x 2^{-L(x)} \leq 1$. (This is called the "Kraft inequality".) *Hint:* First, show that the inequality holds exactly if all paths to leaves have the equal length $\lceil \log_2 k \rceil$.
    3. Prove Eq. E.1.
2. *Shannon/Fano codes* The previous exercise proves that for any coding tree, $\mathbb{E}\left[L(X)\right] \geq H[X]$. But if there was no way to come *close* to $H[X]$, it would be less compelling to use the entropy to measure variability. The following construction shows that there is always a coding tree which does, in fact, come close to the lower bound[18].
    1. Order the possible values of x so that $p(x_1) \leq p(x_2) \leq \ldots \leq p(x_k)$
    2. Find the point in the ordered list of values where, as nearly as possible, the two halves have equal probability. Divide the list there, and make the first question "is X in the left half or the right half of the list?"

[18] The construction is attributed to R. M. Fano by Shannon (1948, §9), but I have not verified the reference.

3. Continue subdividing each group of values for nearly equal probability.
4. Stop when a group consists of just a single value; make this a leaf node.

(Figure [[REF]] shows an example.)

Prove that with this coding tree, $\mathbb{E}[L(X)] \leq 1 + H[X]$, i.e., that Eq. E.3 holds.

3. Prove that $\lim_{x \to 0} x \log_2 x = 0$. *Hints:* use L'Hopital's rule to show that $x \ln x \to 0$ as $x \to 0$.
4. *Entropy is maximized by a uniform distribution*
   1. Show that $H[X]$ has a *local* maximum at a uniform distribution. *Hint:* Treat the probability mass function $p(x)$ as a vector of length $|\mathcal{X}|$, and optimize over the coordinates of the vector. Use a Lagrange multiplier [[REF]] to enforce the constraint $\sum_x p(x) = 1$.
   2. Show that $-H[X]$ is a *convex* function of the probability distribution, i.e., that, for any $a \in [0,1]$ and any two probability mass functions p and q,

$$\sum_{x \in \mathcal{X}} (ap(x) + (1-a)q(x)) \log_2 (ap(x) + (1-a)q(x)) \leq -a \sum_x p(x) \log_2 p(x) - (1-a) \sum_x q(x) \log \tag{E.76}$$

   Hint: Is it enough to show that $x \log_2 x$ is convex?
5. 1. Show that $H[X] \geq 0$. *Hint:* Probabilities are between 0 and 1.
   2. Show that $H[X] = 0$ if, and only if, the distribution of X is degenerate, i.e., $\Pr(X = x^*) = 1$ for some x^*. *Hint:* Same hint.
6. *Independence and the additivity of entropies*
   1. Show that if $X_1, X_2, \ldots X_k$ are all independent, then the joint entropy is the sum of the marginal entropies, $H[X_1, X_2, \ldots X_k] = \sum_{i=1}^k H[X_i]$. *Hint:* $\log_2 \prod_{i=1}^k p_i = \sum_{i=1}^k \log_2 p_i$.
   2. Can you show that if $H[X_1, X_2, \ldots X_k] = \sum_{i=1}^k H[X_i]$, then the X_i are all independent?
7. 1. Prove Eq. E.14.
   2. Use Eq. E.14 to prove Eq. E.15.
   3. Use Eq. E.15 and Exercise 6 to show that $H[Y|X] = H[Y]$ if, and only if, $X \perp\!\!\!\perp Y$.
8. *Many-one functions can preserve mutual information.* Suppose that X is uniformly distributed on the numbers 1 to 8, and Y is either 0 or 1. If X is even, $Y = 1$ with probability 0.75, and if X is odd, $Y = 1$ with probability 0.25.
   1. Calculate $H[X]$, $H[Y]$, $H[Y|X]$ and $I[X;Y]$.
   2. Consider $B = b(X)$, which is 0 when $X \leq 4$ and 1 when $X > 4$. Calculate $H[B]$, $H[Y|B]$ and $I[B;Y]$.
   3. Now consider $T = t(X)$, which is 0 when $X \in \{1,3\}$, 1 when $X \in \{2,4\}$, 3 when $X \in \{5,7\}$ and 4 when $X \in \{6,8\}$. Calculate $H[T]$, $H[Y|T]$ and $I[T;Y]$.
   4. Now consider $S = s(X)$, which is 0 when X is even and 1 when X is odd. Calculate $H[S]$, $H[Y|S]$ and $I[S;Y]$.

5. Consider the following rule: "For any function t, if $t(x_1) = t(x_2)$ implies $\Pr(Y|X = x_1) = \Pr(Y|X = x_2)$, then $I[T;Y] = I[X;Y]$". Can you prove this, or find a counter-example?

9. Prove Eq. E.20. *Hint:* Show that $\sum_y p(x,y)f(x) = p(x)f(x)$, for any function f.
10. Prove Eq. 10. *Hint:*
11. Suppose C is a convex set, and k is any (finite) natural number. Show that for any set of points $x_1, x_2, \ldots x_k \in C$, and any set of weights with $a_i \geq 0$, $\sum_{i=1}^k a_k = 1$, $\sum_{i=1}^k a_i x_i \in C$. *Hint:* Use mathematical induction.
12. 1. Suppose that for any points x_1 and x_2, all the points on the line between $(x_1, f(x_1))$ and $(x_2, f(x_2))$ are at or above the graph of $f(x)$. Take x_1 as fixed, and show that this implies there is some s such that $f(x_2) \geq f(x_1) + s(x_2 - x_1)$, for all x_2.
    2. Fix your favorite x_1, and suppose that there is some s such that, for any x_2, $f(x_2) \geq f(x_1) + s(x_2 - x_1)$. Show that the line between $(x_1, f(x_1))$ and $(x_2, f(x_2))$ must lie on or above the graph of $f(x)$.
13. Suppose that f is differentiable, with derivative f'.
    1. Show that if $f(x_2) \geq f(x_1) + f'(x_1)(x_2 - x_1)$, for arbitrary x_1 and x_2, then f is a convex function.
    2. Show that if f is convex, then $f(x_2) \geq f(x_1) + f'(x_1)(x_2 - x_1)$.
14. Show that if f is a convex function, then for any a, both the set where $f(x) \leq a$ and the set where $f(x) < a$ are convex sets.
15. *Local and global minima of convex functions* Throughout, suppose that f is a convex function, but do not assume that it has only one argument, or that it is differentiable.
    1. Suppose that x is a local minimum of f. Prove that it is also a global minimum on any convex set C containing x. *Hint:* Suppose that there was an $x^* \in C$ with $f(x^*) < f(x)$, and derive a contradiction.
    2. Suppose that the whole domain of f is a convex set. Prove that a local minimum must also be a global minimum.
    3. Suppose that there are (at least) two global minima, x_1 and x_2. Prove that any convex combination of x_1 and x_2 must also be a global minimum.
    4. Prove that if f is *strictly* convex, then the global minimum is unique. *Hint:* Suppose otherwise, and use Exercise 153 to derive a contradiction to the definition of strict convexity.
16. *Local and global maxima of concave functions* Throughout, suppose that f is a concave function, but do not assume that it has only one argument, or that it is differentiable. *Hint:* Do Exercise 15.
    1. Suppose that x is a local maximum of f. Prove that it is also a global maximum on any convex set C containing x.
    2. Suppose that the whole domain of f is a convex set. Prove that a local maximum is also a global maximum.

3. Suppose that f is strictly concave. Prove that the global maximum is unique.

17. *Economic interpretation of concave optimization* Suppose that we are running some economic enterprise which consumes inputs $x_1, x_2, \ldots x_k$, and delivers output $f(x_1, x_2, \ldots x_k)$. Inputs might be things like the labor of workers, electricity, the use of a building, raw materials, etc., and output is measured in physical units — so much wheat, or so many heads of cabbage, or finished chairs, or plastic bags, or for that matter patients treated. Economists say that there are **diminishing marginal returns** when the second derivatives all exist, and the matrix of second derivatives is negative-definite. (This means that more and more inputs are needed to keep increasing the output.)
    1. Show that a local maximum of the output is also a global maximum of the output, so that a necessary and sufficient condition for x to be a global maximum is that $\nabla f(x) = 0$.
    2. Suppose that each unit of output has a value of $p > 0$. Show that a necessary and sufficient condition for finding the global maximum of the value of the output is, still, that $\nabla f(x) = 0$.
    3. Suppose that each input itself has a value $\lambda_1, \lambda_2, \ldots \lambda_k$, all ≥ 0; we can stack these into a vector λ. The **value-added** of the enterprise is therefore $pf(x_1, x_2, \ldots x_k) - \sum_{i=1}^{k} x_i \lambda_i$. Show that a necessary and sufficient condition for x to be a global maximum is that $\nabla f(x) = p^{-1}\lambda$. *Hint:* Eq. E.73.
    4. *Economic interpretation pointedly continued* Suppose that one of the inputs, say x_1, is assigned no value, $\lambda_1 = 0$. How much of it will be consumed when value-added is maximized?
    5. Suppose there are two enterprises, producing outputs valued at p_1 and p_2, represented by two functions f_1 and f_2, both of which exhibit decreasing marginal returns. Show that the necessary and sufficient condition for maximizing the *total* value-added $\nabla f_1(x) = p_1^{-1}\lambda$ and $\nabla f_2(x) = p_2^{-1}\lambda$. Does this generalize to arbitrarily many enterprises?
    6. Suppose that there are no values on inputs, but there are constraints on how much of each input can be used, i.e., for each i, $x_i \leq c_i$. Show that maximizing the value of the output under these constraints has the same form as in Exercises 173 and 175. *Hint:* Lagrange multipliers D.3.2.

    Note: When the values of inputs and outputs are just market prices, maximizing value added is the same as maximizing profit (and Exercise **??** then tells us something about what a profit-maximizing system will do to unpriced resources). However, the same math applies when we try to optimize anything else; as Exercise 176 suggests, prices, or something very like them, are implicit in optimization under constraints. See also the Further Reading above, p. 676.

Appendix F

Relative Distributions and Smooth Tests of Goodness-of-Fit

In §5.2.2.3, we saw how to use the quantile function to turn uniformly-distributed random numbers into random numbers with basically arbitrary distributions. In this chapter, we will look at two closely-related data-analysis tools which go the other way, trying to turn data into uniformly-distributed numbers. One of these, the **smooth test**, turns a lot of problems into ones of testing a uniform distribution. Another, the **relative distribution**, gives us a way of comparing whole distributions, rather than specific statistics (like the expectation or the variance).

F.1 Smooth Tests of Goodness of Fit

F.1.1 From Continuous CDFs to Uniform Distributions

Suppose that X has probability density function f, and that f is continuous. The corresponding cumulative distribution function F is then continuous and strictly increasing (on the support of f). Since F is a fixed function, we can ask what the probability distribution of $F(X)$ is. Clearly,

$$\Pr\left(F(X) \leq 0\right) = 0 \tag{F.1}$$

$$\Pr\left(F(X) \leq 1\right) = 1 \tag{F.2}$$

Since F is continuous and strictly increasing, it has an inverse, the quantile function Q, which is also continuous and strictly increasing. Then, for $0 \leq a \leq 1$,

$$\Pr\left(F(X) \leq a\right) = \Pr\left(Q(F(X)) \leq Q(a)\right) \tag{F.3}$$

$$= \Pr\left(X \leq Q(a)\right) \tag{F.4}$$

$$= F(Q(a)) = a \tag{F.5}$$

Thus, when F is continuous and strictly-increasing, $F(X)$ is uniformly distributed on the unit interval,

$$F(X) \sim \text{Unif}(0,1) \tag{F.6}$$

If the distribution of X is F, but we guess that it has some other distribution, with CDF F_0, then this trick will not work. $F_0(X)$ will still be in the unit interval, but it won't be uniformly distributed:

This only works if X really is distributed according to F. If instead X were

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

distributed according, say, F_0, then $F(X)$ will still be in the unit interval, but it will not be uniformly distributed:

$$\Pr(F_0(X) \leq a) = \Pr(X \leq Q_0(a)) \tag{F.7}$$
$$= F(Q_0(a)) \neq a \tag{F.8}$$

because $F_0 \neq Q^{-1}$.

Putting this together, we see that when X has a continuous distribution, $F(X) \sim \text{Unif}(0,1)$ if and only if F is the cumulative distribution function for X. This means that we can reduce the problem of testing whether $X \sim F$ to that of testing whether $F(X)$ is uniform. We need to work out *one* testing problem, rather than many different testing problems for many different distributions.

F.1.2 Testing Uniformity

Now we have a random variable, say Y, which lives on the unit interval $[0,1]$, and we want to test whether it is uniformly distributed. There are several different ways we could do this. One frequently-used strategy is to use the Kolmogorov-Smirnov test: calculate the K-S distance,

$$d_{KS} = \max_{a \in [0,1]} \left| \widehat{F}_{n,Y}(a) - a \right| \tag{F.9}$$

where $\widehat{F}_{n,Y}(a)$ is the empirical CDF of Y, and look up the appropriate p-value for the K-S test. One could use any other one-sample non-parametric test here, like Cramér-von Mises or Anderson-Darling[1] All of these tests can work quite well in the right circumstances, and they have the advantage of requiring little additional work over and above typing `ks.test` or the like.

F.1.3 Neyman's Smooth Test

There are however two disadvantages of just applying off-the-shelf tests to check uniformity. One is that it turns out that they often do not have very high power. The other, which is in some ways even more serious, is that rejecting the null hypothesis of uniformity doesn't tell you *how* uniformity fails — it doesn't suggest any sort of natural alternative.

As you can guess from my having brought up these points, there is a test which avoids both difficulties, called **Neyman's smooth test**. It works by embedding the uniform distribution on the unit interval in a larger class of alternatives, and then testing the null of uniformity against those alternatives.

The alternatives all have pdfs of the form

$$g(y;\theta) \equiv \begin{cases} \frac{e^{\sum_{j=1}^{d} \theta_j h_j(y)}}{z(\theta)} & 0 \leq y \leq 1 \\ 0 & \text{elsewhere} \end{cases} \tag{F.10}$$

[1] You could even use a χ^2 test, but this would be dumb. Because the χ^2 test requires discrete data, using it means binning continuous values, thereby destroying information, to no good purpose.

where the h_j are carefully chosen functions (see below), and the **normalizing factor** or **partition function** $z(\theta)$ just makes sure the density integrates to 1:

$$z(\theta) \equiv \int_0^1 e^{\sum_{j=1}^d \theta_j h_j(y)} dy \tag{F.11}$$

No matter what functions we pick for the h_j, uniformity corresponds to the choice $\theta = 0$, since then the density is just 1. As we move θ slightly away from 0, the density departs *smoothly* from uniformity; hence the name of the test.

To ensure that everything works out, we need to put some requirements on the functions h_j: they need to be **orthogonal** to each other and to the constant function,

$$\int_0^1 h_j(y) dy = 0 \tag{F.12}$$

$$\int_0^1 h_j(y) h_k(y) dy = 0 \tag{F.13}$$

and **normalized** in magnitude,

$$\int_0^1 h_j^2(y) dy = 1 \tag{F.14}$$

Further details, while practically important, do not matter for the general idea of the test, so I'll put them off to §F.1.3.1.

We can estimate θ by maximum likelihood. Because uniformity corresponds to $\theta = 0$, we can test the hypothesis that $\theta = 0$ against the alternative that $\theta \neq 0$ with a likelihood ratio test. Writing $\ell(\hat{\theta})$ for the log-likelihood under the MLE, and $\ell(0)$ for the log-likelihood under the null, by general results on the likelihood-ratio, under the null, as $n \to \infty$,

$$2(\ell(\hat{\theta}) - \ell(0)) \rightsquigarrow \chi^2_d \tag{F.15}$$

In fact, $\ell(0) = 0$ (why?), so we only need to calculate the log-likelihood under the alternative, and reject uniformity when, and only when, that log-likelihood is large.

Alternatively, and this was Neyman's original recommendation and what is usually meant by his "smooth test", we can calculate the sample mean of each of the h_j,

$$\overline{h_j} = \frac{1}{n} \sum_{i=1}^n h_j(y_i) \tag{F.16}$$

and form the test statistic

$$\Psi^2 = n \sum_{j=1}^d \overline{h_j}^2 \tag{F.17}$$

which also has a χ^2_d distribution under the null.[2]

[2] To appreciate what's going on, notice that $\overline{h_j} \to 0$ under the null, by the law of large numbers.

It can be shown that Neyman's smooth test has, in a certain sense, optimal power against smooth alternatives like this — see Rayner and Best (1989) or Bera and Ghosh (2002) for the gory details. More importantly, for data analysis, when we reject the null hypothesis of uniformity, we have a ready-made alternative to fall back on, namely $g(y;\hat{\theta})$.

To make all this work, we have to pick some "basis functions" h_j, and we need to decide how many of them we want to use, d.

F.1.3.1 Choice of Function Basis

Neyman's original proposal was to use **orthonormal polynomials** for basis functions: h_j would be a polynomial of degree j, which was orthogonal to all the ones before it,

$$\int_0^1 h_j(y)h_k(y)dy = 0 \ \forall k < j \tag{F.18}$$

including the constant "polynomial" $h_0(y) = 1$, and normalized to size 1,

$$\int_0^1 h_j^2(y)dy = 1 \tag{F.19}$$

Since there are $j+1$ coefficients in a polynomial of degree j, and this gives $j+1$ equations, the polynomial is uniquely determined. In fact, there are recursively formulas which let you find the coefficients of h_j from those of the previous polynomials[3]. Figure F.1 shows the first few of these polynomials, and their exponentiated versions (which are what appear in Eq. F.10).

Experience has shown that the specific choice of basis functions doesn't matter as much as ensuring that they are orthonormal. One could, for instance, use $h_j(y) = c_j \cos 2\pi jy$, where c_j is a normalizing constant[4].

F.1.3.2 Choice of Number of Basis Functions

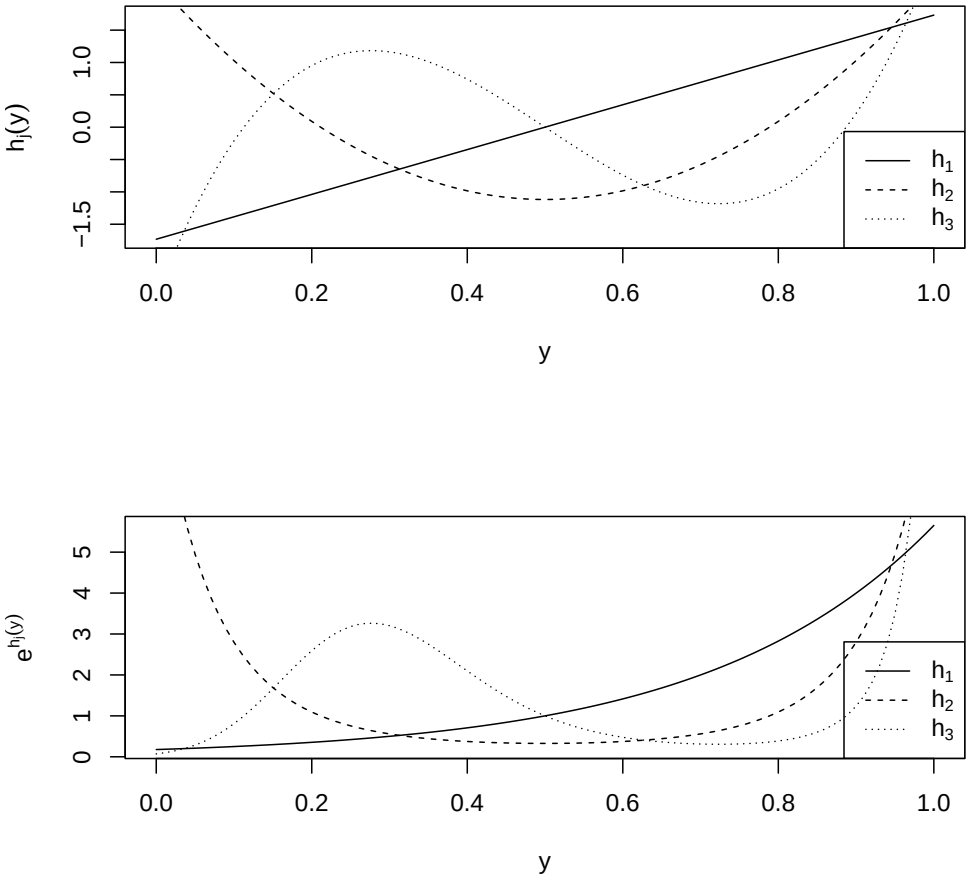
As we make d in Eq. F.10, we include more and more distributions in the alternative to the null hypothesis of uniformity. In fact, since any smooth function on $[0,1]$ can be approximated arbitrarily closely by sufficiently-high order polynomials[5], as we let $d \to \infty$ we eventually get *all* continuous distributions, other than uniformity, as part of the alternative. However, using a large value of d means

(This is where being orthogonal to the constant function $h_0(y) = 1$ comes in.) Multiplying $\overline{h_j}^2$ by n corresponds to looking at $\sqrt{n}\overline{h_j}$, which should, by the central limit theorem, be a Gaussian; the variance of this Gaussian is 1. (This is where normalizing each h_j comes in.) Finally, $\sqrt{n}\overline{h_j}$ and $\sqrt{n}\overline{h_k}$ are uncorrelated. (This is where the mutual orthogonality of the h_j comes in.) Thus, the Ψ^2 statistic is a sum of d uncorrelated standard Gaussians, which has a χ^2_d distribution.

[3] In fact, the polynomials Neyman proposed to use are, as he knew, the "Legendre polynomials", though many math books (and Wikipedia) give the version of those defined on $[-1,1]$, rather than on $[0,1]$. If l_j is the polynomial on $[-1,1]$, then $h_j(y) = l_j(2(y-0.5))$.

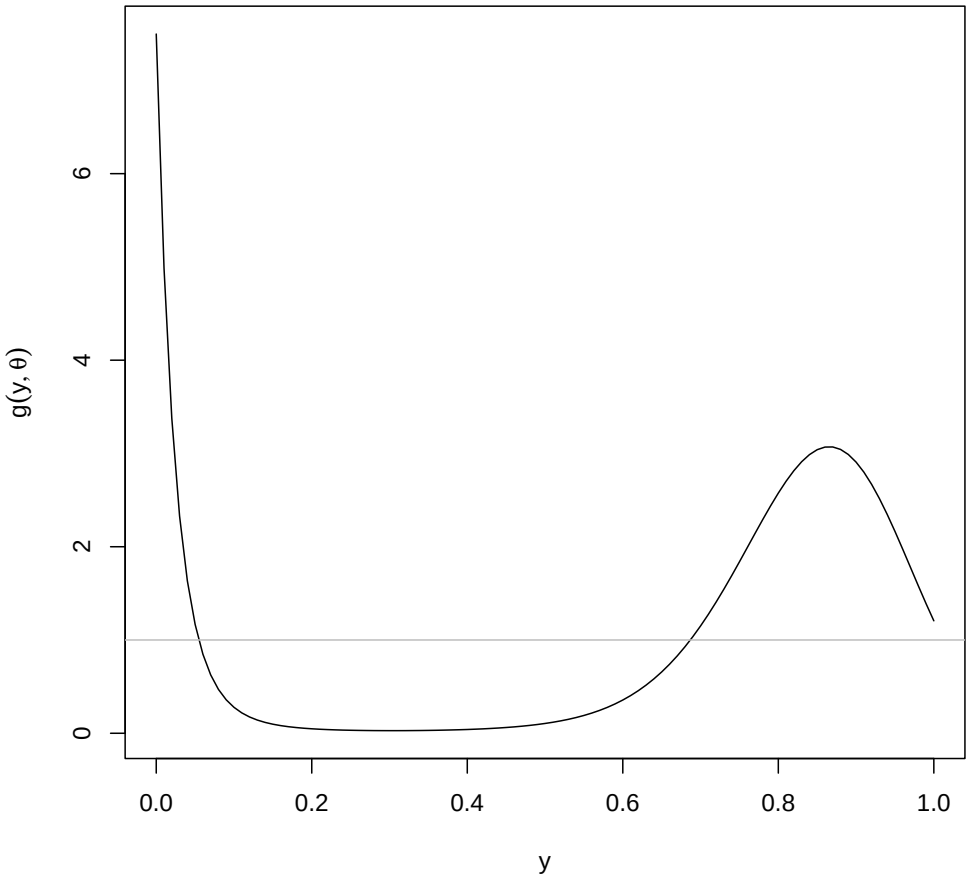
[4] If this makes you think of Fourier analysis, you're right.

[5] This may be obvious, but making it precise (what do we mean by "smooth" and "arbitrarily close"?) is the "Stone-Weierstrauss theorem". There is nothing magic about polynomials here; we could also use sines and cosines, or many other function bases.




```
par(mfrow = c(2, 1))
h1 <- function(y) {
    sqrt(12) * (y - 0.5)
}
h2 <- function(y) {
    sqrt(5) * (6 * (y - 0.5)^2 - 0.5)
}
h3 <- function(y) {
    sqrt(7) * (20 * (y - 0.5)^3 - 3 * (y - 0.5))
}
curve(h1(x), ylab = expression(h[j](y)), xlab = "y")
curve(h2(x), add = TRUE, lty = "dashed")
curve(h3(x), add = TRUE, lty = "dotted")
legend(legend = c(expression(h[1]), expression(h[2]), expression(h[3])), lty = c("solid",
    "dashed", "dotted"), x = "bottomright")
curve(exp(h1(x)), ylab = expression(e^h[j](y)), xlab = "y")
curve(exp(h2(x)), add = TRUE, lty = "dashed")
curve(exp(h3(x)), add = TRUE, lty = "dotted")
legend(legend = c(expression(h[1]), expression(h[2]), expression(h[3])), lty = c("solid",
    "dashed", "dotted"), x = "bottomright")
par(mfrow = c(1, 1))
```

Figure F.1 Left panel: the first three of the basis functions for Neyman's smooth tests, h_1, h_2 and h_3. Each h_j is a polynomial of order j which is orthogonal to the others, in the sense that $\int_0^1 h_j(y)h_k(y)dy = 0$ when $j \neq k$, but normalized in size, $\int_0^1 h_j^2(y)dy = 1$. The right panel shows $e^{h_j(y)}$, to give an indication of how the functions contribute to the probability density in Eq. F.10.




```
x <- (1:1e+06)/1e+06
z <- sum(exp(h1(x) + h2(x) - h3(x)))/1e+06
curve(exp(h1(x) + h2(x) - h3(x))/z, xlab = "y", ylab = expression(g(y, theta)))
abline(h = 1, col = "grey")
```

Figure F.2 Illustration of a smooth alternative density: using the same basis functions as before, with $\theta_1 = 1$, $\theta_2 = 1$, $\theta_3 = -1$. The first two lines of the R calculate the normalizing constant $z(\theta)$ by a simple numerical integral. The grey line shows the uniform density.

estimating a lot of parameters, which means we are at risk of over-fitting. What to do?

Neyman's original advice was to guess a particular value of d before looking

at the data and stick to it. (He thought $d = 4$ would usually be enough.) More modern approaches try to adaptively pick a good value of d. We could attempt this through cross-validation based on the log-likelihood, but what's usually done, in implemented software, is to pick d to maximize Schwarz's information criterion:

$$d^* = \operatorname*{argmax}_d \frac{1}{n}\ell(\widehat{\theta}^{(d)}) - \frac{d}{2}\frac{\log n}{n} \tag{F.20}$$

which imposes an extra penalty for each parameter (d), with the size of the penalty depending on how much data we have, and getting relatively harsher as n grows[6]. So in a **data-driven smooth test** (Kallenberg and Ledwina, 1997), we pick d^* using Eq. F.20, and then compute the test statistic using d^*.

Unfortunately, since d^* is random (through the data), the nice asymptotic theory which says that the test statistic is χ^2_d under the null hypothesis no longer applies. However, this is why we have bootstrapping: by simulating from the null hypothesis, which remember is just $\text{Unif}(0,1)$, and treating the simulation output like real data we can work out the sampling distribution as accurately as we need. This sampling distribution then gives us our p-values.

F.1.3.3 Application: Combining p-Values

One useful property of p-values is that they are always uniformly distributed on $[0,1]$ under the null hypothesis[7]. Suppose we have conducted a bunch of tests of the same null hypothesis — these might be different clinical trials of the same drug, or attempts to replicate some surprising effect in separate laboratories[8]. If the tests are independent, then the p-values should be IID and uniform. It would seem like we should be able to combine these into some over-all p-value. This is *precisely* what Neyman's smooth test of uniformity lets us do.

F.1.3.4 Density Estimation by Series Expansion

As an aside, notice what we have done. By using a large enough d, as I said, densities which look like Eq. F.10 can come as close as we like to any smooth density on $[0,1]$. And now we have at least two ways of picking d: by cross-validation, or by the Schwarz information criterion (Eq. F.20). If we let $d \to \infty$ as $n \to \infty$, then we have a way of approximating any density on the unit interval, without knowing what it was to start with, or assuming a particular parametric form for it. That is, we have a way of doing non-parametric density estimation, at least on $[0,1]$, without using kernels.

If you want to estimate a density on $(-\infty, \infty)$ instead of on $[0,1]$, you can do so

[6] It is common in the literature to see the criterion written out multiplied through by n, or even by $2n$. Also, it is often called the "Bayesian information criterion", or BIC. This is an unfortunate name, because, despite what Schwarz (1978) thought, it really has nothing at all to do with Bayes's rule or even Bayesian statistics. It's best thought of as a fast, but very crude and not always very accurate, approximation to cross-validation. If you want to know more, Claeskens and Hjort (2008) is probably the best reference.

[7] Unless someone has messed up a calculation, that is.

[8] These are typical examples of **meta-analysis**, trying to combine the results of many different data analyses (without just going back to the original data).

by using a transformation, e.g., the inverse logit. This is the opposite of what you did in the homework, where you used a transformation to take $[0,1]$ to $(-\infty,\infty)$ so you could use kernel density estimation.

F.1.4 Smooth Tests of Non-Uniform Parametric Families

Remember that we went into all these details about testing uniformity because we want to test whether X is distributed according to some continuous distribution with CDF F. From §F.1.1, if we define $Y = F(X)$, then $X \sim F$ is equivalent to $Y \sim \text{Unif}(0,1)$, so we have a two-step procedure for testing whether $X \sim F$:

1. Use the CDF F to transform the data, $y_i = F(x_i)$
2. Test whether the transformed data y_i are uniform

Let's think about what the alternatives considered in the test look like. For y, the alternative densities are (to repeat Eq. F.10)

$$g(y;\theta) \equiv \begin{cases} \frac{e^{\sum_{j=1}^{d} \theta_j h_j(y)}}{z(\theta)} & 0 \leq y \leq 1 \\ 0 & \text{elsewhere} \end{cases} \tag{F.21}$$

Since $X = F^{-1}(Y)$, this implies a density for X:

$$g_X(x;\theta) = \frac{e^{\sum_{j=1}^{d} \theta_j h_j(F(x))}}{z(\theta)} \frac{dF}{dx} \tag{F.22}$$

$$= \frac{e^{\sum_{j=1}^{d} \theta_j h_j(F(x))}}{z(\theta)} f(x) \tag{F.23}$$

where f is the pdf corresponding to the CDF F. (Why do we not have to worry about setting this to zero outside some range?) Just like $g(\cdot;\theta)$ is a modulation or distortion of the uniform density, $g_X(\cdot;\theta)$ is a modulation or distortion of $f(\cdot)$. If and when we reject the density f, $g_X(\cdot;\theta)$ is available to us as an alternative.

Even if $h_j(y)$ is a polynomial in y, $h_j(F(x))$ will not (in general) be a polynomial in x, but it remains true that

$$\int_{-\infty}^{\infty} h_j(F(x)) h_k(F(x)) f(x) dx = \delta_{jk} \tag{F.24}$$

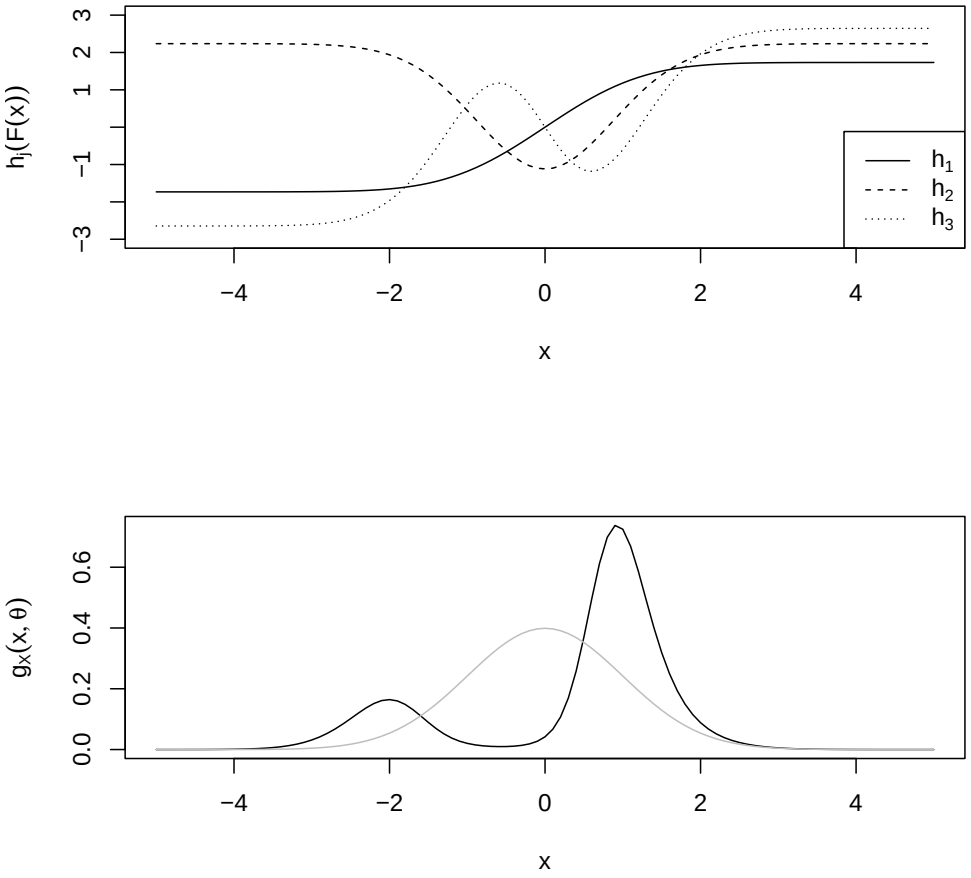
Figure F.3 illustrates what happens to the basis functions, and to particular alternatives.

When it comes to the actual smooth test, we can either use the likelihood ratio, or we can calculate

$$\overline{h_j} = \frac{1}{n} \sum_{i=1}^{n} h_j(y_i) = \frac{1}{n} \sum_{i=1}^{n} h_j(F(x_i)) \tag{F.25}$$

leading as before to the test statistic

$$\Psi^2 = n \sum_{j=1}^{d} \overline{h_j}^2 \tag{F.26}$$




```
par(mfrow = c(2, 1))
curve(h1(pnorm(x)), xlab = "x", ylab = expression(h[j](F(x))), from = -5, to = 5,
    ylim = c(-3, 3))
curve(h2(pnorm(x)), add = TRUE, lty = "dashed")
curve(h3(pnorm(x)), add = TRUE, lty = "dotted")
legend(legend = c(expression(h[1]), expression(h[2]), expression(h[3])), lty = c("solid",
    "dashed", "dotted"), x = "bottomright")
curve(dnorm(x) * exp(h1(pnorm(x)) + h2(pnorm(x)) - h3(pnorm(x)))/z, xlab = "x", ylab = expression(g[
    theta)), from = -5, to = 5)
curve(dnorm(x), add = TRUE, col = "grey")
par(mfrow = c(1, 1))
```

Figure F.3 Left panel: the basis functions from Figure F.1 composed with the standard Gaussian CDF. Right panel: the alternative to the standard Gaussian corresponding to the alternative to the uniform distribution plotted in Figure F.2, i.e., $\theta_1 = \theta_2 = 1$, $\theta_3 = -1$. The grey curve is the standard Gaussian density, corresponding to the flat line in Figure F.2.

The distribution of the test statistics is unchanged under the null hypothesis, i.e., still χ^2_d if d is fixed. (There are still d degrees of freedom, because we are still fixing d parameters from distributions of the form Eq. F.23.) If d is chosen from the data, we still need to bootstrap, but can do so just as before.

F.1.4.1 Estimated Parameters

So far, the discussion has assumed that F is fixed and won't change with the data. This is often not very realistic. Rather, F comes from some parametrized family of distributions, with parameter (say) β, i.e., $F(\cdot;\beta)$ is a different CDF for each value of β. For Gaussians, for instance, β is a vector consisting of the mean and variance (or mean and standard deviation). Let's assume that there are always corresponding densities, $f(\cdot;\beta)$, and these are always continuous.

We don't know β so we have to estimate it. After estimating, we'd like to test whether the model really matches the data. It would be convenient if we could do the following:

1. Get estimate $\widehat{\beta}$ from $x_1, x_2, \ldots x_n$
2. Calculate $y_i = F(x_i; \widehat{\beta})$
3. Apply a smooth test of uniformity to $y_1, y_2, \ldots y_n$

That is, it would be convenient if we could just *ignore* the fact that we had to estimate β.

We can do this if $\widehat{\beta}$ is the maximum likelihood estimate. To understand this, think about the family of alternative distributions we're now considering in the test. Substituting into Eq. F.23, they are

$$g_X(x;\beta,\theta) = \frac{e^{\sum_{j=1}^{d} \theta_j h_j(F(x;\beta))}}{z(\theta)} f(x;\beta) \tag{F.27}$$

The null hypothesis that $X \sim F(\cdot;\beta)$ for some β is thus corresponds to $X \sim G_X(\cdot;\beta,0)$ — we are still fixing d parameters in the larger family. And, generally speaking, when we fix d parameters in a parametric model, we get a χ^2_d distribution in the log-likelihood ratio test. If d is not fixed but data-driven, then, again, we need to bootstrap.

F.1.5 Implementation in R

The main implementation of smooth tests available in R is the `ddst` package (Biecek and Ledwina, 2010), standing for "data-driven smooth tests". It provides a `ddst.uniform.test`, which we could use for any family where we can calculate the CDF. But it also provides functions for directly testing several families of distributions, notably Gaussians (`ddst.norm.test`) and exponentials (`ddst.exp.test`).

F.1.5.1 Some Examples

Let's give `ddst.norm.test` some Gaussian data and see what happens.

```
r <- rnorm(100)
(r.normality <- ddst.norm.test(r))
##
##  Data Driven Smooth Test for Normality
##
## data:  r,   base: ddst.base.legendre,   c: 100
## WT* = 2.3625, n. coord = 1
```

This reminds us what the data was, tells us that the test used Legendre polynomials (as opposed to cosines), that d was selected to be 1, and that the value of the test statistic was 2.36. (The c setting has to do with the order-selection penalty, and is basically ignorable for most users.) These numbers are all attributes of the returned object.

What is missing is the p-value, because this is computationally expensive to calculate. (You can control how many bootstraps it uses, but the default is 1000.)

```
(r.normality <- ddst.norm.test(r, compute.p = TRUE))
##
##  Data Driven Smooth Test for Normality
##
## data:  r,   base: ddst.base.legendre,   c: 100
## WT* = 2.3625, n. coord = 1, p-value = 0.1479
```

So the p-value is 0.148, giving us little reason to reject a Gaussian distribution — which is good, because we're looking at numbers from the standard Gaussian. If we ignored the fact that d was selected from the data and plugged into the corresponding χ^2_d distribution, we'd get a p-value of

```
pchisq(r.normality$statistic, df = 1, lower.tail = FALSE)
##       WT*
## 0.1242842
```

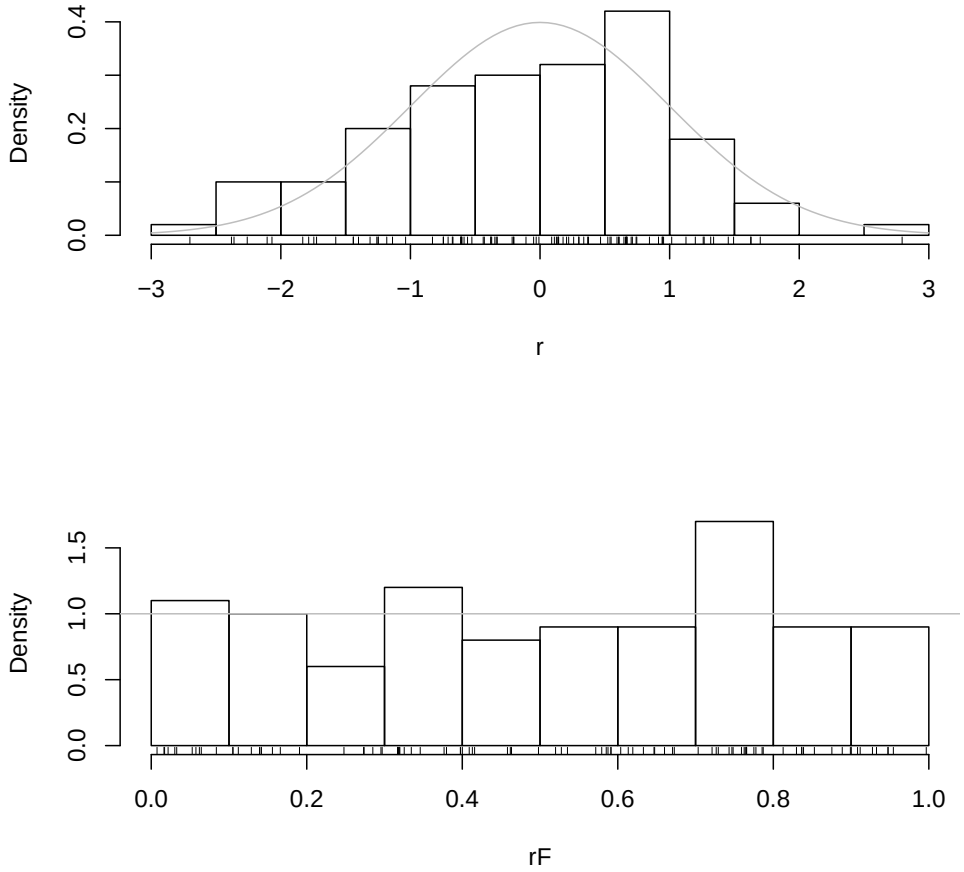
which to say a relative error of about 16%.

What if we give the procedure some non-Gaussian data? Say, the same amount of data from a t distribution with 5 degrees of freedom?

```
ng <- rt(100, df = 5)
ddst.norm.test(ng, compute.p = TRUE)
##
##  Data Driven Smooth Test for Normality
##
## data:  ng,   base: ddst.base.legendre,   c: 100
## WT* = 1.1016, n. coord = 1, p-value = 0.3111
```

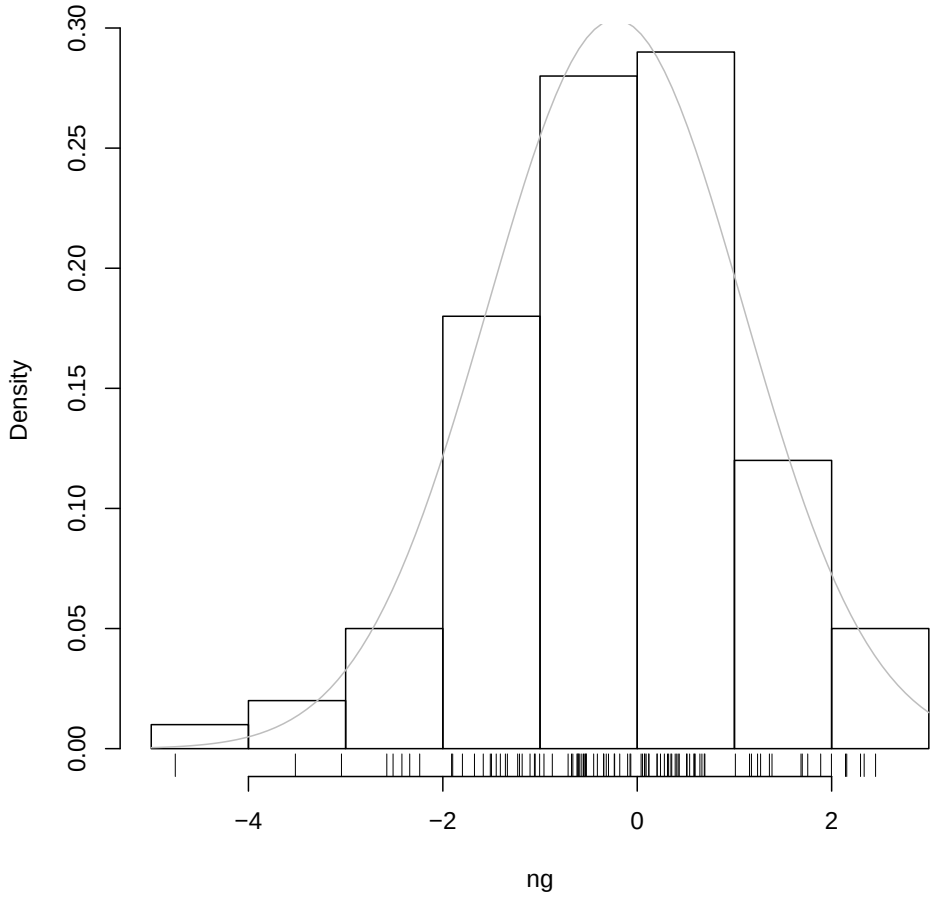
Of course, it won't *always* reject, because the we're only looking at 100 samples, and the t distribution isn't *that* different from a Gaussian. Still, when I repeat this experiment many times, we get quite respectable power at the standard 5% size:

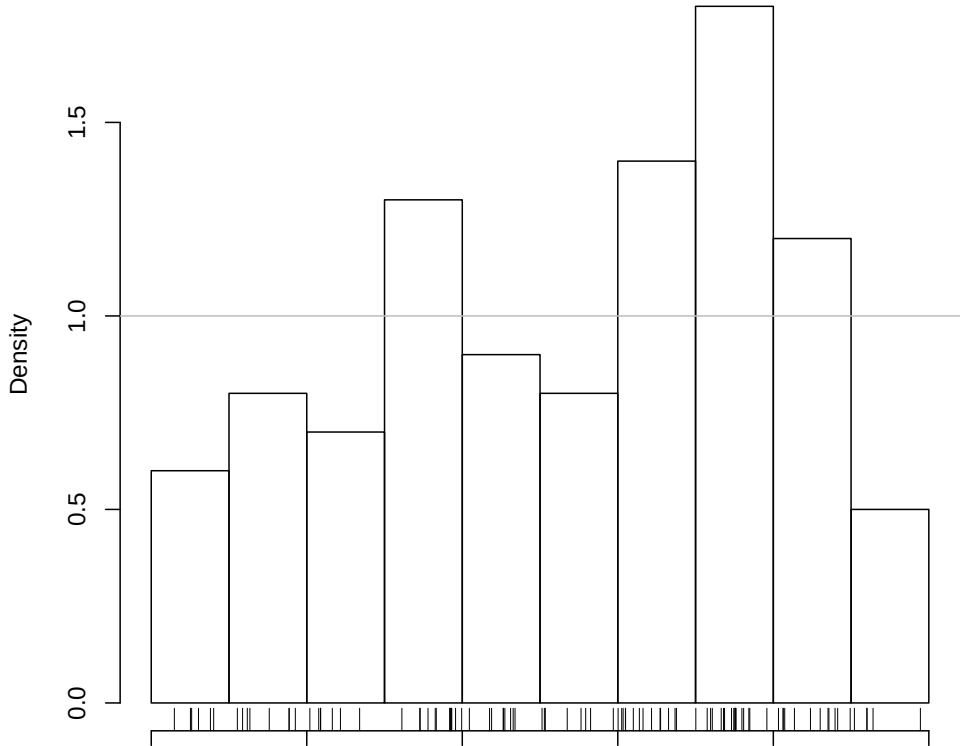
```
mean(replicate(100, ddst.norm.test(rt(100, df = 5), compute.p = TRUE)$p.value) <
    0.05)
## [1] 0.48
```




```
par(mfrow = c(2, 1))
plot(hist(r, plot = FALSE), freq = FALSE, main = "")
rug(r)
curve(dnorm(x), add = TRUE, col = "grey")
rF <- pnorm(r, mean = mean(r), sd = sd(r))
plot(hist(rF, plot = FALSE), freq = FALSE, main = "")
rug(rF)
abline(h = 1, col = "grey")
par(mfrow = c(1, 1))
```

Figure F.4 Left panel: histogram of the 100 random values from the standard Gaussian used in the text (exact values marked along the horizontal axis), plus the true density in grey. Right panel: transforming the data according to the Gaussian *fitted to the data* by maximum likelihood.


Density
0.00
0.05
0.10
0.15
0.20
0.25
0.30
−4
−2
0
2
ng


Density
0.0
0.5
1.0
1.5

See Exercise F.3 for a small project of `ddst.exp.test` to check a Pareto distribution.

F.1.6 Conditional Distributions and Calibration

Suppose that we are not interested in the marginal distribution of X, but rather its conditional distribution given some other variable or variables C (for "covariates"). If the conditional density $f(x|C = c)$ is continuous in x for every c, then it is easy to argue, in parallel with §F.1.1, that $F(X|C = c)$, the conditional CDF, should $\sim \text{Unif}(0, 1)$. So, as long as we use the conditional CDF to transform X, we can apply smooth tests as before.

One important use of this is regression residuals. Suppose X is the target variable of a regression, with C being the predictor variables[9], and we have some parametric distribution in mind for the noise (Gaussian, say), with the noise ϵ being independent of C. Then the model is $X = r(C) + \epsilon$, so looking at the conditional CDF of X given Z is equivalent to looking at the at unconditional CDF of the residuals. We can then actually *test* whether the residuals are Gaussian, rather than just squinting at a Q-Q plot. We could also do this by applying a K-S test to the transformed residuals, but everything that was said above in favor of smooth tests would still apply.

Notice, by the way, that by applying the CDF transformation to the residuals, we are checking whether the model is properly calibrated, i.e., whether events it says happen with probability p actually have a frequency close to p. We do need to impose assumptions about the distribution of the noise to check calibration for a regression model, since if we *just* predict expected values, we say nothing about how often any particular range of values should happen.

Later, when we look at graphical models and at time series, we will see several other important situations where a statistical model is really about conditional distributions, and so can be checked by looking at conditional CDF transformations. It seems to be somewhat more common to apply K-S tests than smooth tests after the conditional CDF transformation (e.g., Bai 2003), but I think this is just because smooth tests are not as widely known and used as they should be.

F.2 Relative Distributions

So far, I have been talking about how we can test whether our data follows some hypothesized distribution, or family of distributions, by using the fact that $F(X)$ is uniform if and only if X has CDF F. If the values of $F(x_i)$ are close enough to being uniform, the true CDF has to be pretty close to F (with high confidence); if they are far from uniform, the true CDF has to be far from F (again with high confidence).

In many situations, however, we already know (or are at least pretty sure) that X doesn't have some distribution, say F_0, and what we are interested in is *how*

[9] I know you're used to X being the predictor and Y being the target.

X fails to follow it; we want, in other words, to compare the distribution of X to some reference distribution F_0. For instance:

1. We are trying a new medical procedure, and we want to compare the distribution of outcomes for patients who got the treatment to those who did not.
2. We want to compare the distribution of some social outcome across two categories at the same time. (For instance, we might compare income, or lifespan, for men and for women.)
3. We might want to compare members of the same category at different times, or in different locations. (We might compare the income distribution of American men in 1980 to that of 2010, or the lifespans of American men in 2010 to those of Canadian men.)
4. We might compare our actual population to the distribution predicted by a model we know to be too simple (or just approximate) to try to learn what it is missing.

You learned how to do comparisons of simple summaries of distributions in baby stats. (For instance, you learned how to compare group means by doing t-tests.) While these certainly have their places, they can miss an awful lot. For example, a few years ago now an anesthesiologist came to the CMU statistics department for help evaluating a new pain-management procedure, which was supposed to reduce how many pain-killers patients recovering from surgery needed. Under both the old procedure and the new one, the distribution was strongly bimodal, with some patients needing very little by way of pain-killers, many needing much more, and a few needing an awful lot of drugs. Simply looking at the change in the mean amount of drugs taken, or even the changes in the mean and the variance, would have told us very little about whether things were any better[10].

In this example, the **reference distribution**, F_0, is given by the distribution of drug demand for patients on the old pain-management protocol. The new or **comparison** sample, $x_1, \ldots x_n$, are realizations of a random variable X, representing the demand for pain-killers under the new protocol. X follows the **comparison distribution** F, which is presumably not the same as F_0; how does it differ?

The idea of the **relative distribution** is to characterize the change in distributions by using F_0 to transform X into $[0, 1]$, and then looking at how it departs from uniformity. The **relative data**, or **grades**, are

$$r_i = F_0(x_i) \tag{F.28}$$

Simply put, we take the comparison data points and see where they fall in the reference distribution.

What is the cumulative distribution function of the relative data? Let's look at this first at the population level, where we have F_0 (the reference distribution)

[10] I am omitting some details, and not providing a reference because the study is still, so far as I know, unpublished.

and F (the comparison distribution), rather than just samples. Let's call the CDF of the relative data G:

$$G(a) \equiv \Pr(R \leq a) \tag{F.29}$$
$$= \Pr(F_0(X) \leq a) \tag{F.30}$$
$$= \Pr(X \leq Q_0(a)) \tag{F.31}$$
$$= F(Q_0(a)) \tag{F.32}$$

where remember $Q_0 = F_0^{-1}$ is the quantile function of the reference distribution. This in turn implies a probability density function of the relative data:

$$g(y) \equiv \left.\frac{dG}{da}\right|_{a=y} \tag{F.33}$$
$$= \left.\frac{dF}{du}\right|_{u=Q_0(y)} \left.\frac{dF_0^{-1}}{da}\right|_{a=y} \tag{F.34}$$
$$= f(Q_0(y)) \frac{1}{f_0(Q_0(y))} = \frac{f(Q_0(y))}{f_0(Q_0(y))} \tag{F.35}$$

This only applies when $y \in [0,1]$; elsewhere, $g(y)$ is straightforwardly 0.

When $g(y) > 1$, we have $f(Q_0(y)) > f_0(Q_0(y))$ — that is, values around $Q_0(y)$ are relatively more probable in the comparison distribution than in the reference distribution. Likewise, when $g(y) < 1$, the comparison distribution puts less weight on values around $Q_0(y)$ than does the reference distribution. If the comparison distribution was exactly the same as the reference distribution, we would, of course, get $g(y) = 1$ everywhere.

One very important property of the relative distribution is that it is invariant under monotone transformations. That is, suppose instead of looking at X, we looked at $h(X)$ for some monotonic function h. (An obvious example would be change of units, but we might also take logs or powers.) Summary statistics like differences in means are generally not even *equi*-variant[11]. But it is easy to check (Exercise F.4) that the relative distribution of $h(X)$ is the same as the relative distribution of X. This expresses the idea that the difference between the reference and comparison distributions is independent of our choice of a coordinate system for X.

F.2.1 Estimating the Relative Distribution

In some situations, the reference distribution can come from a theoretical model, but the comparison distribution is unknown, though we have samples. Estimating the relative density g is then extremely similar to what we had to do in the last

[11] Remember that a statistic, say δ, is a function of the data, $\delta(x_1, x_2, \ldots x_n)$. The statistic is *invariant* under a transformation h if $\delta(h(x_1), h(x_2), \ldots h(x_n)) = \delta(x_1, x_2, \ldots x_n)$ — the transformation does not change the statistic. The statistic is *equivariant* if it "changes along with" the transformation, $\delta(h(x_1), h(x_2), \ldots h(x_n)) = h(\delta(x_1, x_2, \ldots x_n))$. Maximum likelihood estimates are equivariant. Statistics like the mean are equivariant under linear and affine transformations (but not others).

section for hypothesis testing. Non-parametric estimation of g can thus proceed either through fitting series expansions like Eq. F.10 (with a data-driven choice of d, as above), or through using a fixed, data-independent transformation to map $[0, 1]$ to $(-\infty, \infty)$ and using kernel density estimation[12].

If, on the other hand, neither the reference nor the comparison distribution is fully known, but we have samples from both, estimating the relative distribution involves estimating Q_0, the quantile function of the reference distribution. This is typically estimated as just the empirical quantile function, but in principle one could use, say, kernel smoothing to get at Q_0. Once we have an estimate for it, though, we have reduced the problem of estimating g to the case considered in the previous paragraph.

Uncertainty in the estimate of the relative density g is, as usual, most easily assessed through the bootstrap. Be careful to include the uncertainty in estimates of Q_0 as well, if the reference quantiles have to be estimated. One can, however, also use asymptotic approximations (Handcock and Morris, 1999, §9.6).

F.2.2 R Implementation and Examples

Relative distribution methods were introduced by Handcock and Morris (1998, 1999), who also wrote an R package, `reldist`, which is by far the easiest way to work with relative distributions. Rather than explain abstractly how this works, we'll turn immediately to examples.

F.2.2.1 Example: Conservative versus Liberal Brains

Data analysis problem set 18 looks at the data from Kanai *et al.* (2011), which record the volumes of two parts of the brain, the amygdala and the anterior cingulate cortex (ACC), adjusted for body size, sex, etc., and political orientation on a five-point ordinal scale, with 1 being the most conservative and 5 the most liberal[13]. The subjects being British university students, the lowest score for political orientation recorded was 2, and so we will look at relative distributions between those students and the rest of the sample. That is, we take the conservatives as the comparison sample, and the rest as the reference sample[14].

Having loaded the data into the data frame `n90`, we can look at simple density estimates for the two classes and the two variables (Figure F.6). This indicates that conservative subjects tend to have relatively larger amygdalas and relatively smaller ACCs, though with very considerable overlap. (We are not looking at the uncertainty here at all.)

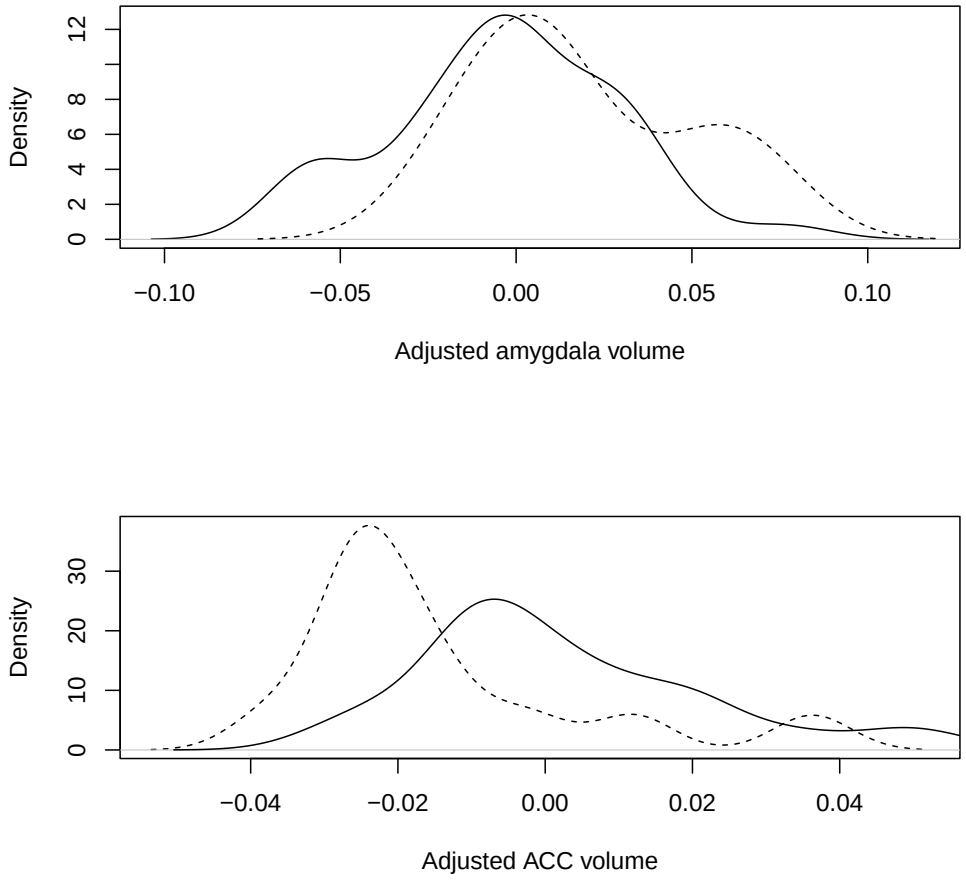
Enough preliminaries; let's find the relative distribution (Figure F.7).

```
acc.rel <- reldist(y = n90$acc[n90$orientation < 3], yo = n90$acc[n90$orientation >
    2], ci = TRUE, yolabs = pretty(n90$acc[n90$orientation > 2]), main = "Relative density of adjust
```

[12] We saw how to do this in the homework

[13] I am grateful to Dr. Kanai for graciously sharing the data.

[14] This implies no value judgment about conservatives being "weird", but rather reflects the fact that there are many fewer of them than of non-conservatives in this data.




```
par(mfrow = c(2, 1))
plot(density(n90$amygdala[n90$orientation > 2]), main = "", xlab = "Adjusted amygdala volume")
lines(density(n90$amygdala[n90$orientation < 3]), lty = "dashed")
plot(density(n90$acc[n90$orientation < 3]), lty = "dashed", main = "", xlab = "Adjusted ACC volume")
lines(density(n90$acc[n90$orientation > 2]))
par(mfrow = c(1, 1))
```

Figure F.6 Estimated densities for the (adjusted) volume of the amygdala (upper panel) and ACC (lower panel) in non-conservative (solid lines) and conservative (dashed) students.

The first argument is the comparison sample; the second is the reference sample. The labeling of the horizontal axis is in terms of the quantiles of the reference distribution; I convert this back to the original units with the optional

`yolabs` argument[15]. The dots show a pointwise 95%-confidence band, but based on asymptotic approximations which should not be taken seriously when there are only 77 reference samples and just 13 comparison samples.

F.2.2.2 Example: Economic Growth Rates

For a second example, let's return to the OECD data on economic growth featured in Chapter 14. We want to know how the economic growth rates of countries which are already economically developed compares to the growth rates of developing and undeveloped countries. I approximate "is a developed country" by "is a membership of the OECD", as in §14.5.1. I will take the non-developed countries as the reference distribution and the OECD members as the comparison group, mostly because there are more of the former and they are more diverse.

The basic commands now go as before (aside from loading the data from a different library):

Examining the resulting plot (Figure F.8), the relative distribution is unimodal, peaking around the 60th percentile of the reference distribution, a growth rate of about 2.5% per year. The relative distribution drops below 1 at both low (negative) or high ($> 0.05\%$) growth rates — developed countries, at least over the period of this data, tend to grow steadily and within a fairly narrow band, without so much of both the positive and negative extremes of non-developed countries[16]

It's also worth illustrating how to use `reldist` for comparison to a theoretical CDF. A *very* primitive, or better yet nihilistic, model of economic growth would say that the factors causing economies to grow or shrink are so many, and so various, and so complicated that there is no hope of tracking them systematic, but rather that we should regard them as effectively random. As we know from introductory probability, the average of many small independent terms has a nearly Gaussian distribution; so we'll just assume that each country grows (or shrinks) by some independent Gaussian amount every year.

Doing this just means applying the cumulative distribution function of the model's distribution to the values from our comparison sample, as in Figure F.9. The result does not look too different from Figure F.8. (This does not mean that the nihilistic model of economic growth is right.)

F.2.3 Adjusting for Covariates

Another nice use of relative distributions is in adjusting for covariates or predictors more flexibly than is easy to do with regression. Suppose that we have

[15] The function `pretty()` is a built-in routine for coming up with reasonable axis tick-marks from a vector. See `help(pretty)`.

[16] It's easy to tell a story for why the distribution of growth rates for poor countries is so wide. Some poor countries grow very slowly or even shrink because they suffer from poor institutions, corruption, war, lack of resources, technological backwardness, etc.; some poor countries grow very quickly if they over-come or escape these obstacles and can quickly make use of technologies developed elsewhere. Nobody has a particular good story for why the growth rates of all developed countries are so similar.

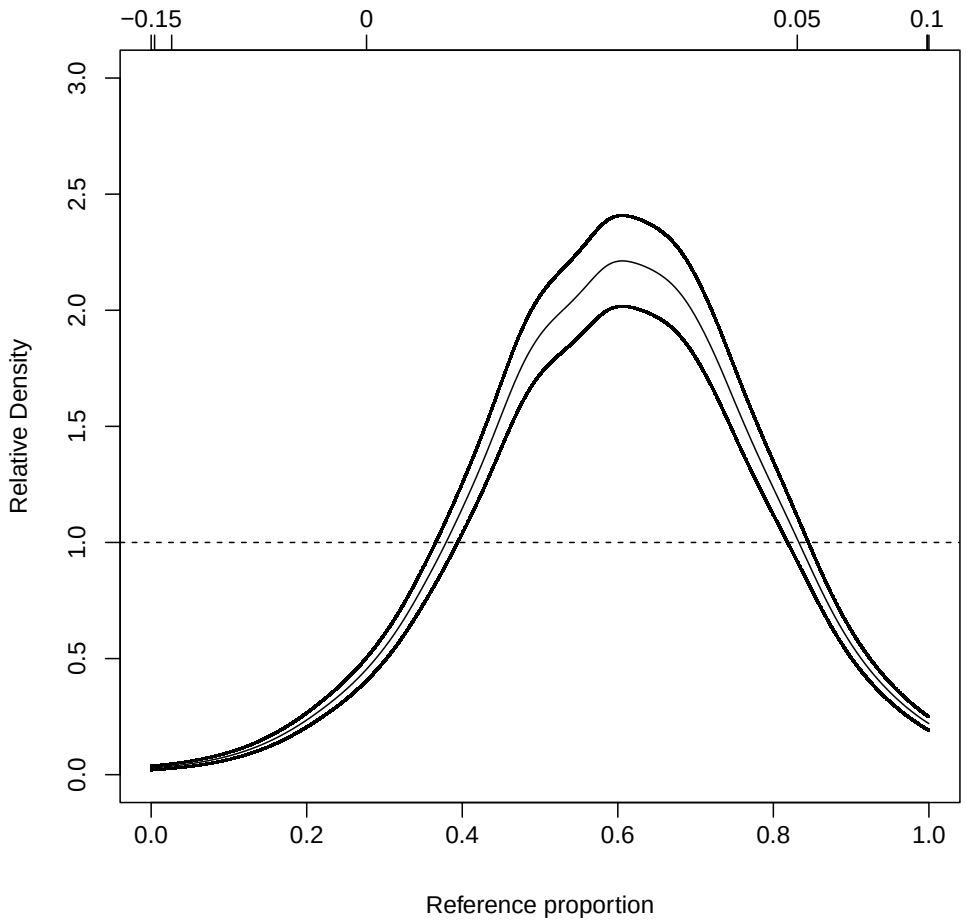
```
##
##    Bayesian density estimation via no-U-turn sampling with
##    a warm-up of size 1000 and 4000 retained samples.
##
## SAMPLING FOR MODEL 'PoissonSimpleMixedModel' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000219 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.19 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 12.0341 seconds (Warm-up)
## Chain 1:                26.1917 seconds (Sampling)
## Chain 1:                38.2258 seconds (Total)
## Chain 1:
```

```
   ## Warning: There were 1 transitions after warmup that exceeded the maximum
                treedepth. Increase max_treedepth above 10. See
      ## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
 ## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing
                          Information was low. See
               ## https://mc-stan.org/misc/warnings.html#bfmi-low
       ## Warning: Examine the pairs() plot to diagnose sampling problems
    ## Warning: The largest R-hat is 1.24, indicating chains have not mixed.
             ## Running the chains for more iterations may help. See
                ## https://mc-stan.org/misc/warnings.html#r-hat
  ## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior
                       means and medians may be unreliable.
             ## Running the chains for more iterations may help. See
               ## https://mc-stan.org/misc/warnings.html#bulk-ess
  ## Warning: Tail Effective Samples Size (ESS) is too low, indicating posterior
                variances and tail quantiles may be unreliable.
             ## Running the chains for more iterations may help. See
               ## https://mc-stan.org/misc/warnings.html#tail-ess
```

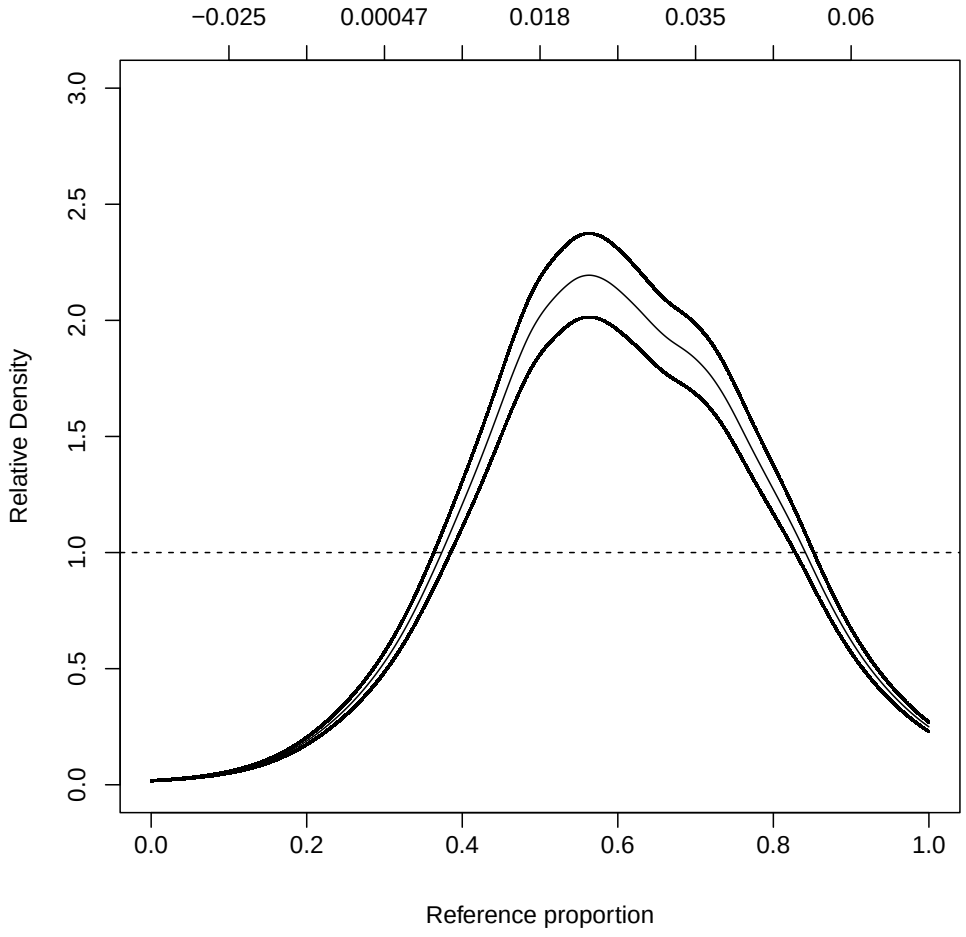
```
##
##    Bayesian density estimation via no-U-turn sampling with
##    a warm-up of size 1000 and 4000 retained samples.
##
## SAMPLING FOR MODEL 'PoissonSimpleMixedModel' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000116 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.16 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 6.54576 seconds (Warm-up)
```

```
##
##    Bayesian density estimation via no-U-turn sampling with
##    a warm-up of size 1000 and 4000 retained samples.
##
## SAMPLING FOR MODEL 'PoissonSimpleMixedModel' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000132 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.32 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 9.53197 seconds (Warm-up)
## Chain 1:                8.59392 seconds (Sampling)
## Chain 1:                18.1259 seconds (Total)
## Chain 1:
```

```
## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing
                        Information was low. See
            ## https://mc-stan.org/misc/warnings.html#bfmi-low
      ## Warning: Examine the pairs() plot to diagnose sampling problems
```



```
##
##    Bayesian density estimation via no-U-turn sampling with
##    a warm-up of size 1000 and 4000 retained samples.
##
## SAMPLING FOR MODEL 'PoissonSimpleMixedModel' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000133 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.33 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 9.83051 seconds (Warm-up)
## Chain 1:                13.6468 seconds (Sampling)
## Chain 1:                23.4773 seconds (Total)
## Chain 1:
```




```
growth.mean <- mean(oecdpanel$growth[!in.oecd])
growth.sd <- sd(oecdpanel$growth[!in.oecd])
```

measurements of *two* variables, X and Z. In general, when we move from the reference population to the comparison population, both variables will change their marginal distributions. If the marginal distribution of Z changes, and the conditional distribution of X given Z did not, then the marginal distribution of X would change. It is often informative to know how the change in the distribution of X compares to what would be anticipated just from the change in Z:

- The two populations might be male and female workers in the same industry, with X income and Z (say) education, or some measure of qualifications.
- The two populations might be students at two different schools, or taught in two different ways, with X their test scores at the end of the year, and Z some measure of prior knowledge.

Write the conditional density of X given Z in the reference population as $f_0(x|z)$. Then, just from the definitions of conditional and marginal probability,

$$f_0(x) = \int f_0(x|z) f_0(z) dz \tag{F.36}$$

If the distribution of the covariate Z is instead taken from the comparison population, we get a *different* distribution for x,

$$f_{0C}(x) = \int f_0(x|z) f(z) dz \tag{F.37}$$

with the C standing for "covariate" or "compensated", depending on who you talk to. This is the distribution we would have seen for X if the distribution of X shifted but the relation between X and Z did not.

Before, we looked at the relative distribution of the comparison distribution F to the reference distribution F_0, which had the density (Eq. F.35) $g(y) = f(Q_0(y))/f_0(Q_0(y))$. Notice that

$$\frac{f(Q_0(y))}{f_0(Q_0(y))} = \frac{f_{0C}(Q_0(y))}{f_0(Q_0(y))} \frac{f(Q_0(y))}{f_{0C}(Q_0(y))} \tag{F.38}$$

The first ratio on the right-hand side the relative density of F_{0C} compared to f_0; the second ratio is the relative density of F compared to F_{0C}.

I have written everything as though Z were just a scalar, but it could be a vector, so we can adjust for multiple covariates at once. Also, it is important to emphasize that there is no implication that Z is in any sense the cause of X here (though such adjustments are often more *interesting* when that's true).

F.2.3.1 Example: Adjusting Growth Rates

Let's look at an example of his this works. The `oecdpanel` data set also includes a variable called `humancap`, which is the log of the average number of years of education of people over the age of fifteen[17]. How do the growth rates of developed countries compare to those of undeveloped countries once we adjust for education?

[17] If you look at `help(oecdpanel)`, it calls this variable "average secondary school enrollment rate", but that's clearly wrong, and examining the original papers referenced there shows the correct meaning of the variable. I am not sure why it was logged. (Incidentally, `humancap` stands for "human

As Figure F.10 shows, after adjusting for education levels, the relative density shifts somewhat to the left, with its peak peaked closer to the median of the reference distribution. That is, some of the higher-than-usual growth of the developed countries can be explained away by their (unusually high: Figure F.11) levels of education. But the relative density is now even *more* sharply peaked than it was before.

Again, it would be rash to read too much causality into this. It could be that education promotes economic growth[18], or it could be that education is a luxury of rich societies, which grow faster than average for other reasons.

F.3 Further Reading

On smooth tests of goodness of fit, see Bera and Ghosh (2002) (a pleasantly *enthusiastic* paper) and Rayner and Best (1989). The `ddst` package is ultimately based on Kallenberg and Ledwina (1997). On relative distributions, see Handcock and Morris (1998) (an expository paper aimed at social scientists) and Handcock and Morris (1999) (a more comprehensive book with technical details).

Exercises

F.1 §F.1.3.1 asserts that one could use cosines orthonormal basis functions in a Neyman test, with $h_j(x) = c_j \cos 2\pi j x$. Find an expression for the normalizing constant c_j such that these functions satisfy Eq. F.18 and Eq. F.19.

F.2 Prove Eq. F.24. *Hint:* change of variables. Also, prove that

$$\int_{-\infty}^{\infty} f(x) \exp^{\sum_{j=1}^{d} \theta_j h_j(F(x))} dx = \int_0^1 \exp^{\sum_{j=1}^{d} \theta_j h_j(y)} dy = z(\theta) \tag{F.39}$$

F.3 If $X \sim \text{Pareto}(\alpha, x_0)$, then $\log X/x_0 \sim \text{Exp}(\alpha)$ — the log of a power-law distributed variable has an exponential distribution. Using the `wealth.dat` data from Chapter 6 and `ddst.exp.test`, test whether net worths over $\$3 \times 10^8$ follow a Pareto distribution.

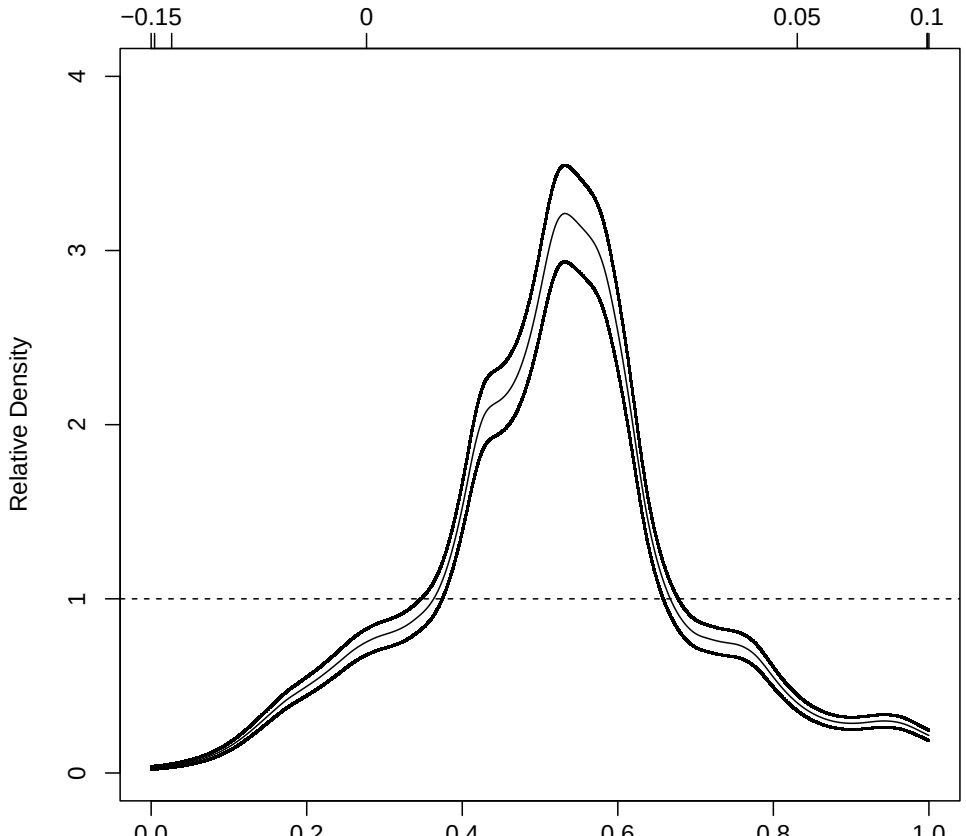
F.4 Let $T = h(X)$ for some fixed and strictly monotonic function h. Prove that the relative density of T is the same as the relative density of X. *Hint:* find the density of T under both the reference and comparison distribution in terms of f_0, f and h.

capital". Whether education is best thought of in this way, or indeed whether years of schooling are a good measure of human capital, are hard questions which we fortunately do not have to answer.)

[18] Certainly it's convenient for a teacher to think so.

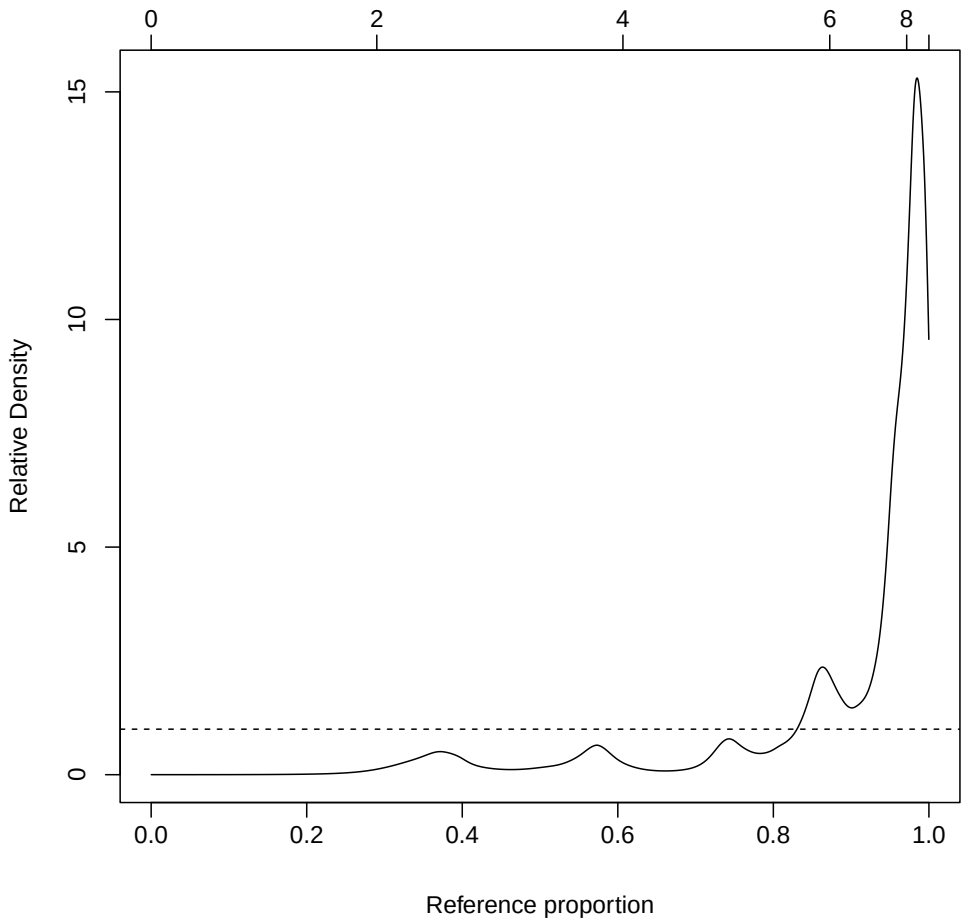
```
##
##    Bayesian density estimation via no-U-turn sampling with
##    a warm-up of size 1000 and 4000 retained samples.
##
## SAMPLING FOR MODEL 'PoissonSimpleMixedModel' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.000132 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.32 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 10.9103 seconds (Warm-up)
## Chain 1:                14.7626 seconds (Sampling)
## Chain 1:                25.6728 seconds (Total)
## Chain 1:
```

```
## Warning: There were 1 chains where the estimated Bayesian Fraction of Missing
                        Information was low. See
            ## https://mc-stan.org/misc/warnings.html#bfmi-low
      ## Warning: Examine the pairs() plot to diagnose sampling problems
 ## Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior
                        means and medians may be unreliable.
          ## Running the chains for more iterations may help. See
            ## https://mc-stan.org/misc/warnings.html#bulk-ess
```



```
##
##    Bayesian density estimation via no-U-turn sampling with
##    a warm-up of size 1000 and 4000 retained samples.
##
## SAMPLING FOR MODEL 'PoissonSimpleMixedModel' NOW (CHAIN 1).
## Chain 1:
## Chain 1: Gradient evaluation took 0.00014 seconds
## Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 1.4 seconds.
## Chain 1: Adjust your expectations accordingly!
## Chain 1:
## Chain 1:
## Chain 1: Iteration:    1 / 5000 [  0%]  (Warmup)
## Chain 1: Iteration:  500 / 5000 [ 10%]  (Warmup)
## Chain 1: Iteration: 1000 / 5000 [ 20%]  (Warmup)
## Chain 1: Iteration: 1001 / 5000 [ 20%]  (Sampling)
## Chain 1: Iteration: 1500 / 5000 [ 30%]  (Sampling)
## Chain 1: Iteration: 2000 / 5000 [ 40%]  (Sampling)
## Chain 1: Iteration: 2500 / 5000 [ 50%]  (Sampling)
## Chain 1: Iteration: 3000 / 5000 [ 60%]  (Sampling)
## Chain 1: Iteration: 3500 / 5000 [ 70%]  (Sampling)
## Chain 1: Iteration: 4000 / 5000 [ 80%]  (Sampling)
## Chain 1: Iteration: 4500 / 5000 [ 90%]  (Sampling)
## Chain 1: Iteration: 5000 / 5000 [100%]  (Sampling)
## Chain 1:
## Chain 1:  Elapsed Time: 33.8186 seconds (Warm-up)
## Chain 1:                131.82 seconds (Sampling)
## Chain 1:                165.638 seconds (Total)
## Chain 1:
```

```
## Warning: There were 7 transitions after warmup that exceeded the maximum
              treedepth. Increase max_treedepth above 10. See
    ## https://mc-stan.org/misc/warnings.html#maximum-treedepth-exceeded
     ## Warning: Examine the pairs() plot to diagnose sampling problems
```



Appendix G

Nonlinear Dimensionality Reduction

PCA (Chapter 15) and factor models (Chapter 16) are examples of *linear* dimension reduction; they're good when there's low-dimensional structure in the data, but that structure is a plane or other linear subspace. *Non*-linear dimension reduction is an obvious extension, but, since there are many ways of being non-linear, it's nowhere near as settled a subject as the linear special case. After a stark example of how linear methods can fail (§G.1), this appendix goes through, in some detail, implementing one particular nonlinear dimension reduction method, called "locally linear embedding", which directly builds on what we've done with PCA (§§G.3–**??**). The further reading (§G.5) points towards other methods, with some indication of their virtues and drawbacks.

G.1 Why We Need Nonlinear Dimensionality Reduction

Consider the points shown in Figure G.1. Even though there are two variables, a.k.a. coordinates, all of the points fall on a one-dimensional curve (as it happens, a logarithmic spiral). This is exactly the kind of constraint which it would be good to recognize and exploit — rather than using two separate coordinates, we could just say how far along the curve a data-point is.

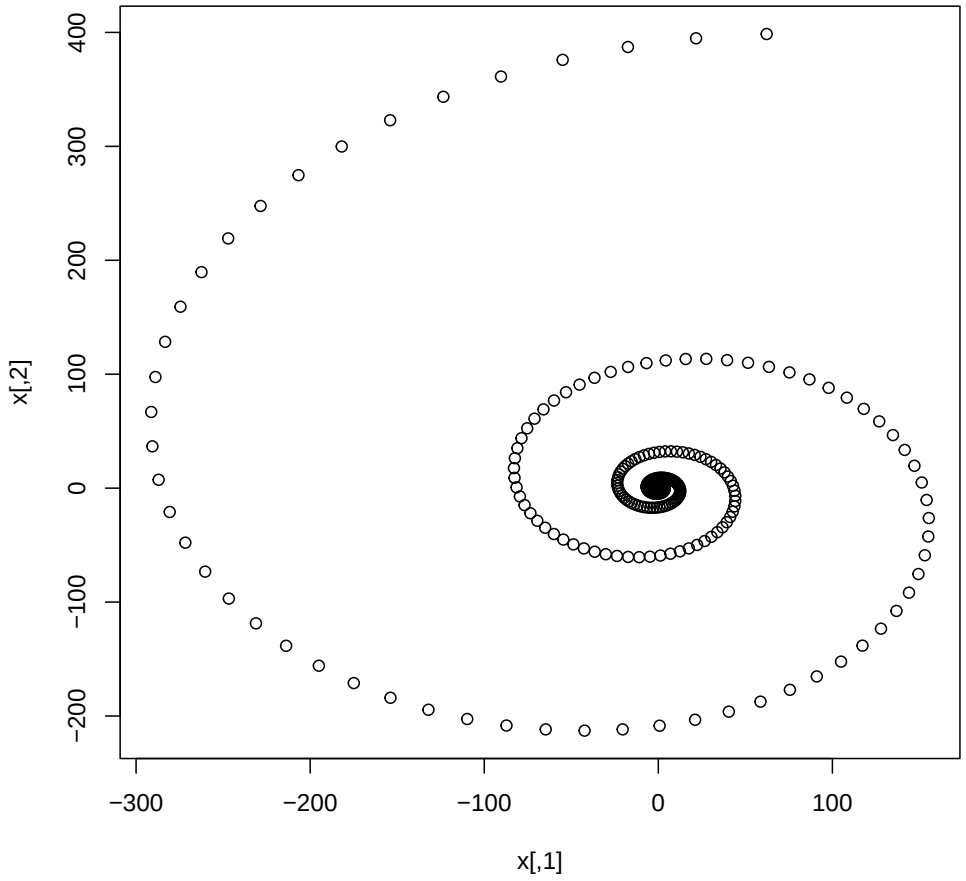
PCA will do poorly with data like this. Remember that to get a one-dimensional representation out of it, we need to take the first principal component, which is the *straight line* along which the data's projections have the most variance. If this works for capturing structure along the spiral, then projections on to the first PC should have the same order that the points have along the spiral.[1] Since, fortuitously, the data are already in that order, we can just plot the first PC against the row index (Figure G.3). The results are — there is really no other word for it — *screwy*.

So, PCA with one principal component fails to capture the one-dimensional structure of the spiral. We could add another principal component, but then we've just rotated our two-dimensional data. In fact, *any* linear dimensionality-reduction method is going to fail here, simply because the spiral is not even approximately a one-dimensional *linear subspace*.

What then are we to do?

[1] It wouldn't matter if the coordinate increased as we went out along the spiral or decreased, just so long as it was monotonic.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

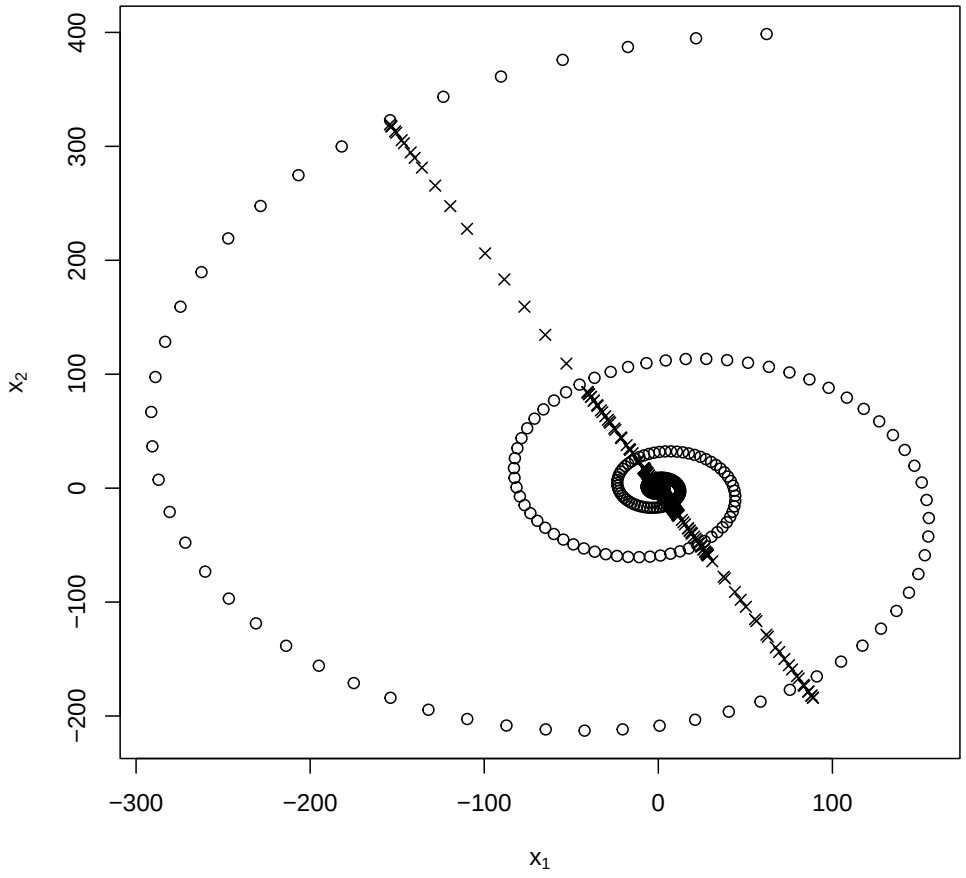



```
x = matrix(c(exp(-0.2 * (-(1:300)/10)) * cos(-(1:300)/10), exp(-0.2 * (-(1:300)/10)) *
    sin(-(1:300)/10)), ncol = 2)
plot(x)
```

Figure G.1 Two-dimensional data constrained to a smooth one-dimensional region, namely the logarithmic spiral, $r = e^{-0.2\theta}$ in polar coordinates.

1. Stick to not-too-nonlinear structures.
2. Somehow decompose nonlinear structures into linear subspaces.
3. Generalize the eigenvalue problem of minimizing distortion.

There's not a great deal to be said about (1). Some curves can be approximated by linear subspaces without too much heartbreak. (For instance, see Figure G.4.)

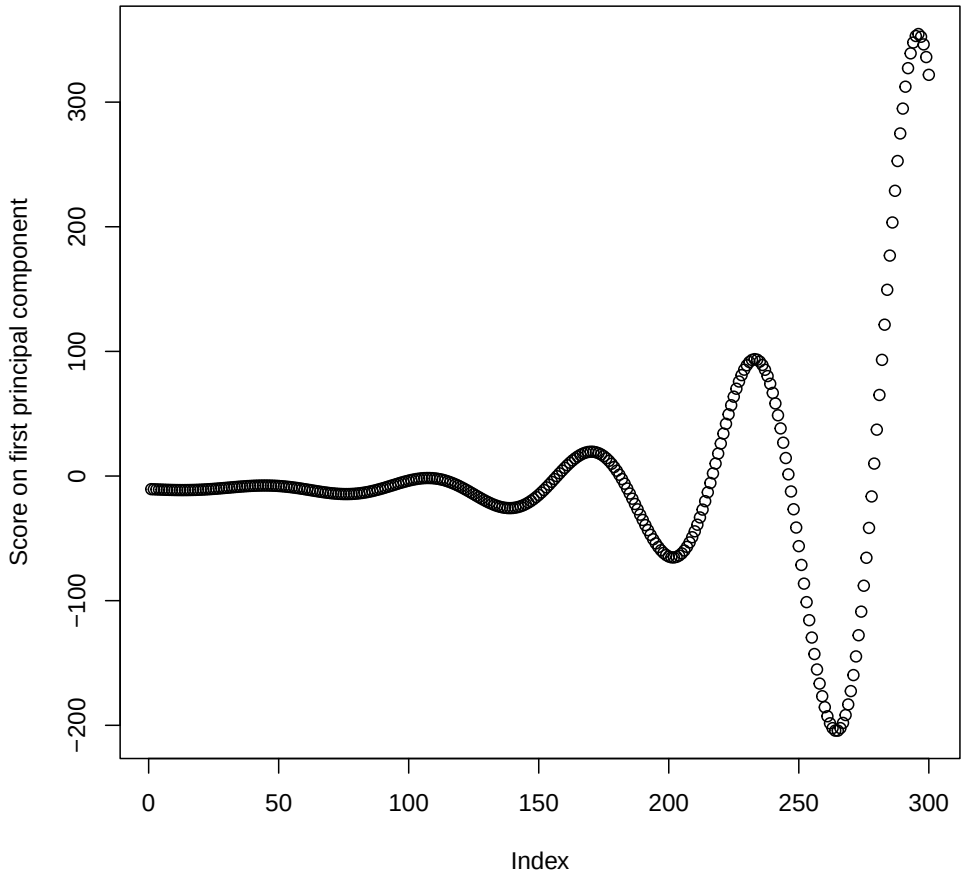



```
fit.all <- prcomp(x)
approx.all <- fit.all$x[, 1] %*% t(fit.all$rotation[, 1])
plot(x, xlab = expression(x[1]), ylab = expression(x[2]))
points(approx.all, pch = 4)
```

Figure G.2 Spiral data (circles) replotted with their one-dimensional PCA approximations (crosses).

We can use things like PCA on them, and so long as we remember that we're just seeing an approximation, we won't go too far wrong. But fundamentally this is *weak*. (2) is hoping that we can somehow build a strong method out of this weak one; as it happens we can, and it's called locally linear embedding (and its variants). The last is diffusion maps.

TODO: Include material or rewrite

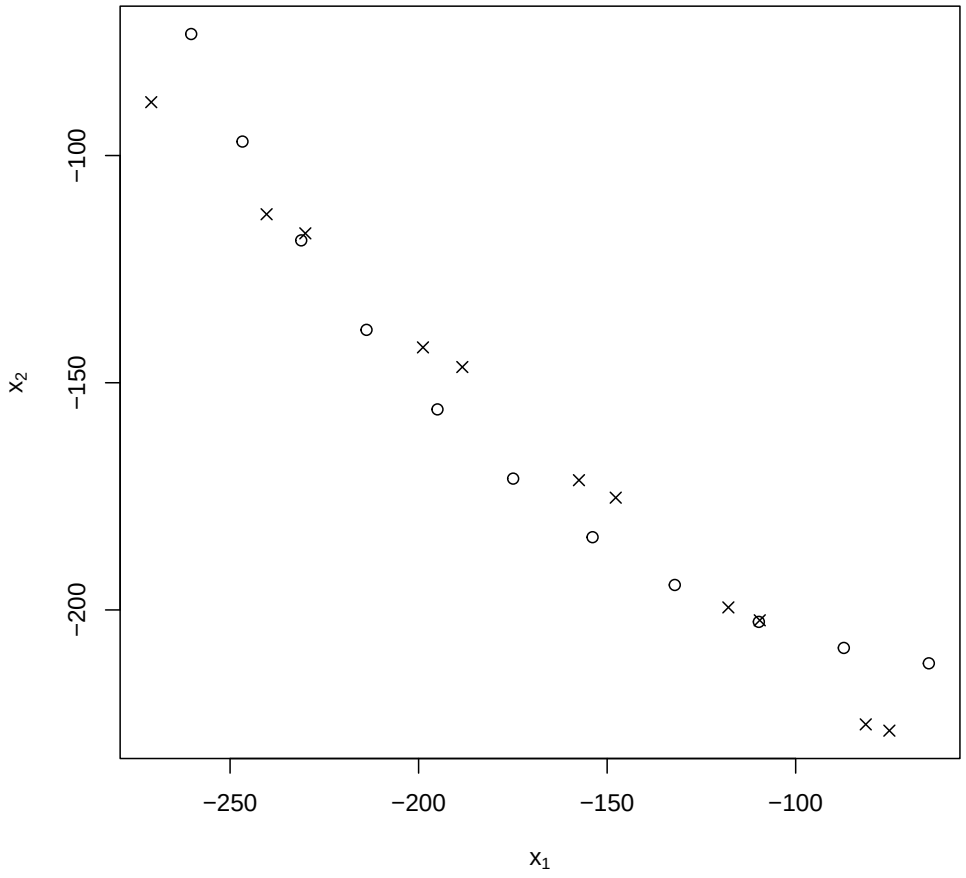



```
plot(fit.all$x[, 1], ylab = "Score on first principal component")
```

Figure G.3 Projections of the spiral points on to their first principal component.

G.2 Local Linearity and Manifolds

Let's look again at Figure G.4. A one-dimensional linear subspace is, in plain words, a straight line. By doing PCA on this part of the data alone, we are approximating a segment of the spiral curve by a straight line. Since the segment is not *very* curved, the approximation is reasonably good. (Or rather, the segment was chosen so the approximation would be good, consequently it had to have low curvature.) Notice that this error is not a random scatter of points around the




```
fit = prcomp(x[270:280, ])
pca.approx = fit$x[, 1] %*% t(fit$rotation[, 1]) + colMeans(x[270:280, ])
plot(rbind(x[270:280, ], pca.approx), type = "n", xlab = expression(x[1]), ylab = expression(x[2]))
points(x[270:280, ])
points(pca.approx, pch = 4)
```

Figure G.4 Portion of the spiral data (circles) together with *its* one-dimensional PCA approximation (crosses).

line, but rather a systematic mis-match between the true curve and the line — a **bias** which would not go away no matter how much data we had from the spiral. The size of the bias depends on how big a region we are using, and how much the tangent direction to the curve changes across that region — the average

curvature. By using small regions when the curvature is high and big regions when the curvature is low, we can maintain any desired degree of approximation.

If we shifted to a different part of the curve, we could do PCA on the data there, too, getting a different principal component and a different linear approximations to the data. Generally, as we move around the degree of curvature will change, so the size of the region we'd use would also need to grow or shrink.

This suggests that we could make some progress towards learning nonlinear structures in the data by patching together lots of linear structures. We could, for example, divide up the whole data space into regions, and do a separate PCA in each region. Here we'd hope that in each region we needed only a single principal component. Such hopes would generally be dashed, however, because this is a bit too simple-minded to really work.

1. We'd need to chose the number of regions, introducing a trade-off between having many points in each region (so as to deal with noise) and having small regions (to keep the linear approximation good).
2. Ideally, the regions should be of different sizes, depending on average curvature, but we don't know the curvature.
3. What happens at the boundaries between regions? The principal components of adjacent regions could be pointing in totally different directions.

Nonetheless, this is the core of a good idea. To make it work, we need to say just a little about **differential geometry**, specifically the idea of a **manifold**.[2] For our purposes, a manifold is a smooth, curved subset of a Euclidean space, in which it is **embedded**. The spiral curve (not the isolated points I plotted) is a one-dimensional manifold in the plane, just as are lines, circles, ellipses and parabolas. The surface of a sphere or a torus is a two-dimensional manifold, like a plane. The essential fact about a q-dimensional manifold is that it can be arbitrarily well-approximated by a q-dimensional linear subspace, the **tangent space**, by taking a sufficiently small region about any point.[3] (This generalizes the fact any sufficiently small part of a curve about any point looks very much like a straight line, the tangent line to the curve at that point.) Moreover, as we move from point to point, the local linear approximations change continuously, too. The more rapid the change in the tangent space, the bigger the curvature of the manifold. (Again, this generalizes the relation between curves and their tangent lines.) So if our data come from a manifold, we should be able to do a local linear

[2] Differential geometry is a very beautiful and important branch of mathematics, with its roots in the needs of geographers in the 1800s to understand the curved surface of the Earth in detail (**geodesy**). The theory of curved spaces they developed for this purpose generalized the ordinary vector calculus and Euclidean geometry, and turned out to provide the mathematical language for describing space, time and gravity (Einstein's general theory of relativity; Lawrie (1990)), the other fundamental forces of nature (gauge field theory; Lawrie (1990)), dynamical systems Arnol'd (1973); Guckenheimer and Holmes (1983), and indeed statistical inference (information geometry; Kass and Vos (1997); Amari and Nagaoka (1993/2000)). Good introductions are Spivak (1965) and Schutz (1980) (which confines the physics to one (long) chapter on applications).

[3] If it makes you happier: every point has an open neighborhood which is homeomorphic to $\mathbb{R}^q$, and the transition from neighborhood to neighborhood is continuous and differentiable.

approximation around every part of the manifold, and then smoothly interpolate them together into a single global system. To do dimensionality reduction — to learn the manifold — we want to find these global low-dimensional coordinates.[4]

G.3 Locally Linear Embedding (LLE)

Locally linear embedding (or: local linear embedding, you see both) is a clever scheme for finding low-dimensional global coordinates when the data lie on (or very near to) a manifold embedded in a high-dimensional space. The trick is to do a different linear dimensionality reduction at each point (because locally a manifold looks linear) and then combine these with minimal discrepancy. It was introduced by Roweis and Saul (2000), though Saul and Roweis (2003) has a fuller explanation. I don't think it uses any elements which were unknown, mathematically, since the 1950s. Rather than diminishing its inventors achievement, this should make the rest of us feel humble...

The LLE procedure has three steps: it builds a neighborhood for each point in the data; finds the weights for linearly approximating the data in that neighborhood; and finally finds the low-dimensional coordinates best reconstructed by those weights. This low-dimensional coordinates are then returned.

To be more precise, the LLE algorithm is given as inputs an $n \times p$ data matrix $\mathbf{X}$, with rows $\vec{x}_i$; a desired number of dimensions $q < p$; and an integer k for finding local neighborhoods, where $k \geq q + 1$. The output is supposed to be an $n \times q$ matrix $\mathbf{Y}$, with rows $\vec{y}_i$.

1. For each $\vec{x}_i$, find the k nearest neighbors.
2. Find the weight matrix $\mathbf{w}$ which minimizes the residual sum of squares for reconstructing each $\vec{x}_i$ from its neighbors,
$$RSS(\mathbf{w}) \equiv \sum_{i=1}^{n} \|\vec{x}_i - \sum_{j \neq i} w_{ij}\vec{x}_j\|^2 \tag{G.1}$$
where $w_{ij} = 0$ unless $\vec{x}_j$ is one of $\vec{x}_i$'s k-nearest neighbors, and for each i, $\sum_j w_{ij} = 1$. (I will come back to this constraint below.)
3. Find the coordinates $\mathbf{Y}$ which minimize the reconstruction error using the weights,
$$\Phi(\mathbf{Y}) \equiv \sum_{i=1}^{n} \|\vec{y}_i - \sum_{j \neq i} w_{ij}\vec{y}_j\|^2 \tag{G.2}$$

[4] There are technicalities here which I am going to gloss over, because this is not a book on differential geometry. (Read one, it's good for you!) The biggest one is that most manifolds don't admit of a truly *global* coordinate system, one which is good everywhere without exception. But the places where it breaks down are usually isolated point and easily identified. For instance, if you take a sphere, almost every point can be identified by latitude and longitude — except for the poles, where longitude becomes ill-defined. Handling this in a mathematically precise way is tricky, but since these are probability-zero cases, we can ignore them in a statistics class.

subject to the constraints that $\sum_i Y_{ij} = 0$ for each j, and that $\mathbf{Y}^T\mathbf{Y} = \mathbf{I}$. (I will come back to those constraints below, too.)

G.3.1 Finding Neighborhoods

In step 1, we define local neighborhoods for each point. By defining these in terms of the k nearest neighbors, we make them physically large where the data points are widely separated, and physically small when the density of the data is high. We don't *know* that the curvature of the manifold is low when the data are sparse, but we do know that, whatever is happening out there, we have very little idea what it is, so it's safer to approximate it crudely. Conversely, if the data are dense, we can capture both high and low curvature. If the actual curvature is low, we might have been able to expand the region without loss, but again, this is playing it safe. So, to summarize, using k-nearest neighborhoods means we take a fine-grained view where there is a lot of data, and a coarse-grained view where there is little data.

It's not strictly necessary to use k-nearest neighbors here; the important thing is to establish *some* neighborhood for each point, and to do so in a way which conforms or adapts to the data.

G.3.2 Finding Weights

Step 2 can be understood in a number of ways. Let's start with the local linearity of a manifold. Suppose that the manifold was *exactly* linear around $\vec{x}_i$, i.e., that it and its neighbors belonged to a q-dimensional linear subspace. Since $q+1$ points in generally define a q-dimensional subspace, there would be *some* combination of the neighbo rs which reconstructed $\vec{x}_i$ exactly, i.e., some set of weights w_{ij} such that

$$\vec{x}_i = \sum_j w_{ij}\vec{x}_j \tag{G.3}$$

Conversely, if there are such weights, then $\vec{x}_i$ and (some of) its neighbors *do* form a linear subspace. Since every manifold is locally linear, by taking a sufficiently small region around each point we get arbitrarily close to having these equations hold — $n^{-1}RSS(\mathbf{w})$ should shrink to zero as n grows.

Vitally, the *same* weights would work to reconstruct $\mathbf{x}_i$ both in the high-dimensional embedding space and the low-dimensional subspace. This means that it is the weights around a given point which characterize what the manifold looks like there (provided the neighborhood is small enough compared to the curvature). Finding the weights gives us the same information as finding the tangent space. This is why, in the last step, we will only need the weights, not the original vectors.

Now, about the constraints that $\sum_j w_{ij} = 1$. This can be understood in two ways, geometrically and probabilistically. Geometrically, what it gives us is invariance under translation. That is, if we add any vector $\vec{c}$ to $\vec{x}_i$ and all of its

neighbors, nothing happens to the function we're minimizing:

$$\vec{x}_i + \vec{c} - \sum_j w_{ij}(\vec{x}_j + \vec{c}) = \vec{x}_i + \vec{c} - \left(\sum_j w_{ij}\vec{x}_j\right) - \vec{c} \tag{G.4}$$

$$= \vec{x}_i - \sum_j w_{ij}\vec{x}_j \tag{G.5}$$

Since we are looking at the same shape of manifold no matter how we move it around in space, translational invariance is a constraint we want to impose.

Probabilistically, forcing the weights to sum to one makes $\mathbf{w}$ a stochastic transition matrix.[5] This should remind you of page-rank, where we built a Markov chain transition matrix from the graph connecting web-pages. There is a tight connection here, which we'll return to next time under the heading of **diffusion maps**; for now this is just to tantalize.

We will see below how to actually minimize the squared error computationally; as you probably expect by now, it reduces to an eigenvalue problem. Actually it reduces to a bunch (n) of eigenvalue problems: because there are no constraints *across* the rows of $\mathbf{w}$, we can find the optimal weights for each point separately. Naturally, this simplifies the calculation.

G.3.2.1 $k > p$

If k, the number of neighbors, is greater than p, the number of variables, then (in general) the space spanned by k distinct vectors is the whole space. Then $\vec{x}_i$ can be written *exactly* as a linear combination of its k-nearest neighbors.[6] In fact, if $k > p$, then not only is there a solution to $\vec{x}_i = \sum_j w_{ij}\vec{j}$, there are generally infinitely many solutions, because there are more unknowns (k) than equations (p). When this happens, we say that the optimization problem is **ill-posed**, or **irregular**. There are many ways of **regularizing** ill-posed problems. A common one, for this case, is what is called L_2 or **Tikhonov** regularization: instead of minimizing

$$\|\vec{x}_i - \sum_j w_{ij}\vec{x}_j\|^2 \tag{G.6}$$

pick an $\alpha > 0$ and minimize

$$\|\vec{x}_i - \sum_j w_{ij}\vec{x}_j\|^2 + \alpha \sum_j w_{ij}^2 \tag{G.7}$$

This says: pick the weights which minimize a combination of reconstruction error *and* the sum of the squared weights. As $\alpha \to 0$, this gives us back the least-squares problem. To see what the second, sum-of-squared-weights term does, take the opposite limit, $\alpha \to \infty$: the squared-error term becomes negligible, and we just

[5] Actually, it really only does that if $w_{ij} \geq 0$. In that case we are approximating $\vec{x}_i$ not just by a linear combination of its neighbors, but by a **convex** combination. Often one gets all positive weights anyway, but it can be helpful to impose this extra constraint.

[6] This is easiest to see when $\vec{x}_i$ lies inside the body which has its neighbors as vertices, their **convex hull**, but is true more generally.

want to minimize the Euclidean ("L_2") norm of the weight vector w_{ij}. Since the weights are constrained to add up to 1, we can best achieve this by making all the weights equal — so some of them can't be vastly larger than the others, and they stabilize at a definite preferred value. Typically α is set to be small, but not zero, so we allow some variation in the weights if it really helps improve the fit.

We will see how to actually implement this regularization later, when we look at the eigenvalue problems connected with LLE. The L_2 term is an example of a **penalty term**, used to stabilize a problem where just matching the data gives irregular results, and there is an art to optimally picking λ; in practice, however, LLE results are often fairly insensitive to it, when it's needed at all[7]. Remember, the whole situation only comes up when $k > p$, and p can easily be very large — in gene-expression data analysis we often have thousands of variables from each measurement.

G.3.3 Finding Coordinates

As I said above, if the local neighborhoods are small compared to the curvature of the manifold, weights in the embedding space and weights on the manifold should be the same. (More precisely, the two sets of weights are exactly equal for linear subspaces, and for other manifolds they can be brought arbitrarily close to each other by shrinking the neighborhood sufficiently.) In the third and last step of LLE, we have just calculated the weights in the embedding space, so we take them to be *approximately* equal to the weights on the manifold, and solve for coordinates on the manifold.

So, taking the weight matrix $\mathbf{w}$ as fixed, we ask for the $\mathbf{Y}$ which minimizes

$$\Phi(\mathbf{Y}) = \sum_i \left\| \vec{y}_i - \sum_{j \neq i} \vec{y}_j w_{ij} \right\|^2 \tag{G.8}$$

That is, what should the coordinates $\vec{y}_i$ be on the manifold, that *these* weights reconstruct them?

As mentioned, some constraints are going to be needed. Remember that we saw above that we could add any constant vector $\vec{c}$ to $\vec{x}_i$ and its neighbors without affecting the sum of squares, because $\sum_j w_{ij} = 1$. We could do the same with the $\vec{y}_i$, so the minimization problem, as posed, has an infinity of equally-good solutions. To fix this — to "break the degeneracy" — we impose the constraint

$$\frac{1}{n} \sum_i \vec{y}_i = 0 \tag{G.9}$$

Since if the mean vector was *not* zero, we could just subtract it from all the $\vec{y}_i$ without changing the quality of the solution, this is just a book-keeping convenience.

[7] It's no accident that the scaling factor for the penalty term is written with a Greek letter; it can also be seen as the Lagrange multiplier enforcing a constraint on the solution (§D.3.3).

Similarly, we also impose the convention that

$$\frac{1}{n}\mathbf{Y}^T\mathbf{Y} = \mathbf{I} \tag{G.10}$$

i.e., that the covariance matrix of $\mathbf{Y}$ be the (q-dimensional) identity matrix. This is not as substantial as it looks. If we found a solution where the covariance matrix of $\mathbf{Y}$ was *not* diagonal, we could use PCA to rotate the new coordinates on the manifold so they were uncorrelated, giving a diagonal covariance matrix. The only bit of this which is not, again, a book-keeping convenience is assuming that all the coordinates have the same variance — that the diagonal covariance matrix is in fact $\mathbf{I}$.

This optimization problem is like multi-dimensional scaling (p. 364): we are asking for low-dimensional vectors which preserve certain relationships (averaging weights) among high-dimensional vectors. We are also asking to do it under constraints, which we will impose through Lagrange multipliers. Once again, it turns into an eigenvalue problem, though one just a bit more subtle than what we saw with PCA in Chapter 15[8].

Unfortunately, finding the coordinates does *not* break up into n smaller problems, the way finding the weights did, because each row of $\mathbf{Y}$ appears in Φ multiple times, once as the focal vector $\vec{y}_i$, and then again as one of the neighbors of other vectors.

G.3.4 More Fun with Eigenvalues and Eigenvectors

To sum up: for each $\vec{x}_i$, we want to find the weights w_{ij} which minimize

$$RSS_i(\mathbf{w}) = \|\vec{x}_i - \sum_j w_{ij}\vec{x}_j\|^2 \tag{G.11}$$

where $w_{ij} = 0$ unless $\vec{x}_j$ is one of the k nearest neighbors of $\vec{x}_i$, under the constraint that $\sum_j w_{ij} = 1$. Given those weights, we want to find the q-dimensional vectors $\vec{y}_i$ which minimize

$$\Phi(\mathbf{Y}) = \sum_{i=1}^{n} \|\vec{y}_i - \sum_j w_{ij}\vec{y}_j\|^2 \tag{G.12}$$

with the constraints that

$$n^{-1}\sum_i \vec{y}_i = 0 \tag{G.13}$$

$$n^{-1}\mathbf{Y}^T\mathbf{Y} = \mathbf{I}\ . \tag{G.14}$$

[8] One reason to suspect the appearance of eigenvalues, in addition to my very heavy-handed foreshadowing, is that eigenvectors are automatically orthogonal to each other and normalized, so making the columns of $\mathbf{Y}$ be the eigenvectors of some matrix would automatically satisfy Eq. G.10.

G.3.5 Finding the Weights

In this subsection, assume that j just runs over the neighbors of $\vec{x}_i$, so we don't have to worry about the weights (including w_{ii}) which we know are zero.

We saw that RSS_i is invariant if we add an arbitrary $\vec{c}$ to all the vectors. Set $\vec{c} = -\vec{x}_i$, centering the vectors on the focal point $\vec{x}_i$:

$$RSS_i = \| \sum_j w_{ij}(\vec{x}_j - \vec{x}_i)\|^2 \tag{G.15}$$

$$= \| \sum_j w_{ij}\vec{z}_j\|^2 \tag{G.16}$$

defining $\vec{z}_j = \vec{x}_j - \vec{x}_i$. If we correspondingly define the $k \times p$ matrix $\mathbf{z}$, and set $\mathbf{w}_i$ to be the $k \times 1$ matrix, the vector we get from the sum is just $\mathbf{w}_i^T\mathbf{z}$. The squared magnitude of any vector $\vec{r}$, considered as a row matrix $\mathbf{r}$, is $\mathbf{r}\mathbf{r}^T$, so

$$RSS_i = \mathbf{w}_i^T\mathbf{z}\mathbf{z}^T\mathbf{w}_i \tag{G.17}$$

Notice that $\mathbf{z}\mathbf{z}^T$ is a $k \times k$ matrix consisting of all the inner products of the neighbors. This symmetric matrix is called the **Gram matrix** of the set of vectors, and accordingly abbreviated $\mathbf{G}$ — here I'll say $\mathbf{G}_i$ to remind us that it depends on our choice of focal point $\vec{x}_i$.

$$RSS_i = \mathbf{w}_i^T\mathbf{G}_i\mathbf{w}_i \tag{G.18}$$

Notice that the data matter only in so far as they determine the Gram matrix $\mathbf{G}_i$; the problem is invariant under any transformation which leaves all the inner products alone (translation, rotation, mirror-reversal, etc.).

We want to minimize RSS_i, but we have the constraint $\sum_j w_{ij} = 1$. We impose this via a Lagrange multiplier, λ.[9] To express the constraint in matrix form, introduce the $k \times 1$ matrix of all 1s, call it $\mathbf{1}$.[10] Then the constraint has the form $\mathbf{1}^T\mathbf{w}_i = 1$, or $\mathbf{1}^T\mathbf{w}_i - 1 = 0$. Now we can write the Lagrangian:

$$\mathcal{L}(\mathbf{w}_i, \lambda) = \mathbf{w}_i^T\mathbf{G}_i\mathbf{w}_i - \lambda(\mathbf{1}^T\mathbf{w} - 1) \tag{G.19}$$

Taking derivatives, and remembering that G_i is symmetric,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_i} = 2\mathbf{G}_i\mathbf{w}_i - \lambda\mathbf{1} = 0 \tag{G.20}$$

$$\frac{\partial \mathcal{L}}{\partial \lambda} = \mathbf{1}^T\mathbf{w}_i - 1 = 0 \tag{G.21}$$

or

$$\mathbf{G}_i\mathbf{w}_i = \frac{\lambda}{2}\mathbf{1} \tag{G.22}$$

If the Gram matrix is invertible,

$$\mathbf{w}_i = \frac{\lambda}{2}\mathbf{G}_i^{-1}\mathbf{1} \tag{G.23}$$

[9] This λ should not be confused with the penalty-term λ used when $k > p$ (§G.3.5.1).
[10] This should not be confused with the identity matrix, $\mathbf{I}$.

where λ can be adjusted to ensure that everything sums to 1.

G.3.5.1 $k > p$

If $k > p$, we modify the objective function to be

$$\mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i + \alpha \mathbf{w}_i^T \mathbf{w}_i \tag{G.24}$$

where $\alpha > 0$ determines the degree of regularization. Proceeding as before to impose the constraint,

$$\mathcal{L} = \mathbf{w}_i^T \mathbf{G}_i \mathbf{w}_i + \alpha \mathbf{w}_i^T \mathbf{w}_i - \lambda(\mathbf{1}^T \mathbf{w}_i - 1) \tag{G.25}$$

where now λ is the Lagrange multiplier. Taking the derivative with respect to $\mathbf{w}_i$ and setting it to zero,

$$2\mathbf{G}_i \mathbf{w}_i + 2\alpha \mathbf{w}_i = \lambda \mathbf{1} \tag{G.26}$$

$$(\mathbf{G}_i + \alpha \mathbf{I})\mathbf{w}_i = \frac{\lambda}{2}\mathbf{1} \tag{G.27}$$

$$\mathbf{w}_i = \frac{\lambda}{2}(\mathbf{G}_i + \alpha \mathbf{I})^{-1}\mathbf{1} \tag{G.28}$$

where, again, we pick λ to properly normalize the right-hand side.

G.3.6 Finding the Coordinates

As with PCA, it's easier to think about the $q = 1$ case first; the general case follows similar lines. So $\vec{y}_i$ is just a single scalar number, y_i, and $\mathbf{Y}$ reduces to an $n \times 1$ column of numbers. We'll revisit $q > 1$ at the end.

The objective function is

$$\Phi(\mathbf{Y}) = \sum_{i=1}^{n} \left(y_i - \sum_j w_{ij} y_j \right)^2 \tag{G.29}$$

$$= \sum_{i=1}^{n} y_i^2 - y_i \left(\sum_j w_{ij} y_j \right) - \left(\sum_j w_{ij} y_j \right) y_i + \left(\sum_j w_{ij} y_j \right)^2 \tag{G.30}$$

$$= \mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T(\mathbf{w}\mathbf{Y}) - (\mathbf{w}\mathbf{Y})^T\mathbf{Y} + (\mathbf{w}\mathbf{Y})^T(\mathbf{w}\mathbf{Y}) \tag{G.31}$$

$$= ((\mathbf{I} - \mathbf{w})\mathbf{Y})^T((\mathbf{I} - \mathbf{w})\mathbf{Y}) \tag{G.32}$$

$$= \mathbf{Y}^T(\mathbf{I} - \mathbf{w})^T(\mathbf{I} - \mathbf{w})\mathbf{Y} \tag{G.33}$$

Define the $m \times m$ matrix $\mathbf{M} = (\mathbf{I} - \mathbf{w})^T(\mathbf{I} - \mathbf{w})$.

$$\Phi(\mathbf{Y}) = \mathbf{Y}^T\mathbf{M}\mathbf{Y} \tag{G.34}$$

This looks promising — it's the same sort of quadratic form that we maximized in doing PCA.

Now let's use a Lagrange multiplier μ to impose the constraint that $n^{-1}\mathbf{Y}^T\mathbf{Y} = \mathbf{I}$ — but, since $q = 1$, that's the 1×1 identity matrix, i.e., the scalar number 1.

$$\mathcal{L}(\mathbf{Y}, \mu) = \mathbf{Y}^T\mathbf{M}\mathbf{Y} - \mu(n^{-1}\mathbf{Y}^T\mathbf{Y} - 1) \tag{G.35}$$

Note that this μ is not the same as the μ which constrained the weights!

Proceeding as we did with PCA,

$$\frac{\partial \mathcal{L}}{\partial \mathbf{Y}} = 2\mathbf{M}\mathbf{Y} - 2\mu n^{-1}\mathbf{Y} = 0 \tag{G.36}$$

or

$$\mathbf{M}\mathbf{Y} = \frac{\mu}{n}\mathbf{Y} \tag{G.37}$$

so $\mathbf{Y}$ must be an eigenvector of $\mathbf{M}$. Because $\mathbf{Y}$ is defined for each point in the data set, it is a function of the data-points, and we call it an **eigenfunction**, to avoid confusion with things like the eigenvectors of PCA (which are p-dimensional vectors in the space of observables). Because we are trying to minimize $\mathbf{Y}^T\mathbf{M}\mathbf{Y}$, we want the eigenfunctions going with the smallest eigenvalues — the **bottom** eigenfunctions — unlike the case with PCA, where we wanted the **top** eigenvectors.

$\mathbf{M}$ being an $n \times n$ matrix, it has, in general, n eigenvalues, and n mutually orthogonal eigenfunctions. The eigenvalues are real and non-negative; the smallest of them is always zero, with eigenfunction $\mathbf{1}$. To see this, notice that $\mathbf{w}\mathbf{1} = \mathbf{1}$.[11] Then

$$(\mathbf{I} - \mathbf{w})\mathbf{1} = 0 \tag{G.38}$$

$$(\mathbf{I} - \mathbf{w})^T(\mathbf{I} - \mathbf{w})\mathbf{1} = 0 \tag{G.39}$$

$$\mathbf{M}\mathbf{1} = 0 \tag{G.40}$$

Since this eigenfunction is constant, it doesn't give a useful coordinate on the manifold. To get our first coordinate, then, we need to take the two bottom eigenfunctions, and discard the constant.

Again as with PCA, if we want to use $q > 1$, we just need to take multiple eigenfunctions of M. To get q coordinates, we take the bottom $q + 1$ eigenfunctions, discard the constant eigenfunction with eigenvalue 0, and use the others as our coordinates on the manifold. Because the eigenfunctions are orthogonal, the no-covariance constraint is automatically satisfied. Notice that adding another coordinate just means taking another eigenfunction of the *same* matrix $\mathbf{M}$ — as is the case with PCA, but not with factor analysis.

(What happened to the mean-zero constraint? Well, we can add another Lagrange multiplier ν to enforce it, but the constraint is linear in $\mathbf{Y}$, it's $\mathbf{a}\mathbf{Y} = 0$ for some matrix $\mathbf{a}$ (Exercise G.2), so when we take partial derivatives we get

$$\frac{\partial \mathcal{L}(\mathbf{Y}, \mu, \nu)}{\partial \mathbf{Y}} = 2\mathbf{M}\mathbf{Y} - 2\mu\mathbf{Y} - \nu\mathbf{a} = 0 \tag{G.41}$$

and this is the *only* equation in which ν appears. So we are actually free to pick any ν we like, and may as well set it to be zero. Geometrically, this is the translational invariance yet again. In optimization terms, the size of the Lagrange multiplier tells us about how much better the solution could be if we relaxed the

[11] Each row of $\mathbf{w}\mathbf{1}$ is a weighted average of the other rows of $\mathbf{1}$. But all the rows of $\mathbf{1}$ are the same.

```
lle <- function(x, q, k = q + 1, alpha = 0.01) {
    stopifnot(q > 0, q < ncol(x), k > q, alpha > 0)
    kNNs = find.kNNs(x, k)
    w = reconstruction.weights(x, kNNs, alpha)
    coords = coords.from.weights(w, q)
    return(coords)
}
```

CODE EXAMPLE 39: *Locally linear embedding in R. Notice that this top-level function is very simple, and mirrors the math exactly.*

```
find.kNNs <- function(x, k, ...) {
    x.distances = dist(x, ...)
    x.distances = as.matrix(x.distances)
    kNNs = smallest.by.rows(x.distances, k + 1)
    return(kNNs[, -1])
}
```

CODE EXAMPLE 40: *Finding the k nearest neighbors of all the row-vectors in a data frame. As the main text says, this is a fairly slow way to do it, and shouldn't be used on large data sets.*

constraint — when it's zero, as here, it means that the constrained optimum is *also* an unconstrained optimum — but we knew that already!)

G.4 Implementation

Let's break this down from the top. The nice thing about doing this is that the over-all function is four lines, one of which is just the return (Example 39).

G.4.1 Finding the Nearest Neighbors

The following approach is straightforward (exploiting an R utility function, `order`), but not recommended for "industrial strength" uses. A *lot* of thought has been given to efficient algorithms for finding nearest neighbors, and this isn't even close to the state of the art [[cites]]. For large n, the difference in efficiency would be quite substantial. For the present, however, this will do.

To find the k nearest neighbors of each point, we first need to calculate the distances between all pairs of points. The neighborhoods only depend on these distances, not the actual points themselves. We just need to find the k smallest entries in each row of the distance matrix (Example 40).

Most of the work is done either by `dist`, a built-in function optimized for calculating distance matrices, or by `smallest.by.rows` (Example 41), which we are about to write. The $+1$ and -1 in the last two lines come from simplifying that.

`smallest.by.rows` uses the utility function `order`. Given a vector, it returns the *permutation* that puts the vector into increasing order, i.e., its return is a

```
smallest.by.rows <- function(m, k) {
    stopifnot(ncol(m) >= k)
    row.orders = t(apply(m, 1, order))
    k.smallest = row.orders[, 1:k]
    return(k.smallest)
}
```

Code Example 41: *Finding which columns contain the smallest entries in each row.*

vector of integers as long as its input.[12] The first line of `smallest.by.rows` applies `order` to each row of the input matrix `m`. The first column of `row.orders` now gives the column number of the smallest entry in each row of `m`; the second column, the second smallest entry, and so forth. By taking the first k columns, we get the set of the smallest entries in each row. `find.kNNs` applies this function to the distance matrix, giving the indices of the closest points. However, every point is closest to itself, so to get k *neighbors*, we need the $k+1$ closest *points*; and we want to discard the first column we get back from `smallest.by.rows`.

Let's check that we're getting sensible results from the parts.

```
(r <- matrix(c(7, 3, 2, 4), nrow = 2))
##      [,1] [,2]
## [1,]    7    2
## [2,]    3    4
smallest.by.rows(r, 1)
## [1] 2 1
smallest.by.rows(r, 2)
##      [,1] [,2]
## [1,]    2    1
## [2,]    1    2
```

Since $7 > 2$ but $3 < 4$, this is correct. Now try a small distance matrix, from the first five points on the spiral:

```
round(as.matrix(dist(x[1:5, ])), 2)
##      1    2    3    4    5
## 1 0.00 0.11 0.21 0.32 0.43
## 2 0.11 0.00 0.11 0.22 0.33
## 3 0.21 0.11 0.00 0.11 0.22
## 4 0.32 0.22 0.11 0.00 0.11
## 5 0.43 0.33 0.22 0.11 0.00
smallest.by.rows(as.matrix(dist(x[1:5, ])), 3)
##   [,1] [,2] [,3]
## 1    1    2    3
## 2    2    1    3
## 3    3    2    4
## 4    4    3    5
## 5    5    4    3
```

Notice that the first column, as asserted above, is saying that every point is closest to itself. But the two nearest neighbors are right.

[12] There is a lot of control over ties, but we don't care about ties. See `help(order)`, though, it's a handy function.

```
reconstruction.weights <- function(x, neighbors, alpha) {
    stopifnot(is.matrix(x), is.matrix(neighbors), alpha > 0)
    n = nrow(x)
    stopifnot(nrow(neighbors) == n)
    w = matrix(0, nrow = n, ncol = n)
    for (i in 1:n) {
        i.neighbors = neighbors[i, ]
        w[i, i.neighbors] = local.weights(x[i, ], x[i.neighbors, ], alpha)
    }
    return(w)
}
```

CODE EXAMPLE 42: *Iterative (and so not really recommended) function to find linear least-squares reconstruction weights.*

```
local.weights <- function(focal, neighbors, alpha) {
    stopifnot(nrow(focal) == 1, ncol(focal) == ncol(neighbors))
    k = nrow(neighbors)
    neighbors = t(t(neighbors) - focal)
    gram = neighbors %*% t(neighbors)
    weights = try(solve(gram, rep(1, k)))
    if (identical(class(weights), "try-error")) {
        weights = solve(gram + alpha * diag(k), rep(1, k))
    }
    weights = weights/sum(weights)
    return(weights)
}
```

CODE EXAMPLE 43: *Find the weights for approximating a vector as a linear combination of the rows of a matrix.*

```
find.kNNs(x[1:5, ], 2)
##   [,1] [,2]
## 1    2    3
## 2    1    3
## 3    2    4
## 4    3    5
## 5    4    3
```

Success!

G.4.2 Calculating the Weights

First, the slow iterative way (Example 42). Aside from sanity-checking the inputs, this just creates a square, $n \times n$ weight-matrix $\mathbf{w}$, initially populated with all zeroes, and then fills each line of it by calling a to-be-written function, `local.weights` (Example 43).

For testing, it would really be better to break `local.weights` up into two sub-parts — one which finds the Gram matrix, and another which solves for the weights — but let's just test it altogether this once.

```
matrix(mapply("*", local.weights(x[1, ], x[2:3, ], 0.01), x[2:3, ]), nrow = 2)
##           [,1]       [,2]
## [1,]  2.014934 -0.4084473
## [2,] -0.989357  0.3060440
colSums(matrix(mapply("*", local.weights(x[1, ], x[2:3, ], 0.01), x[2:3, ]), nrow = 2))
## [1]  1.0255769 -0.1024033
colSums(matrix(mapply("*", local.weights(x[1, ], x[2:3, ], 0.01), x[2:3, ]), nrow = 2)) -
    x[1, ]
## [1]  0.0104723155 -0.0005531495
```

The `mapply` function is another of the `lapply` family of utility functions. Just as `sapply` sweeps a function along a vector, `mapply` sweeps a multi-argument function (hence the `m`) along multiple argument vectors, recycling as necessary. Here the function is multiplication, so we're getting the products of the reconstruction weights and the vectors. (I re-organize this into a matrix for comprehensibility.) Then I add up the weighted vectors, getting something that looks reasonably close to `x[1,]`. This is confirmed by actually subtract the latter from the approximation, and seeing that the differences are small for both coordinates.

This didn't use the regularization; let's turn it on and see what happens.

```
colSums(matrix(mapply("*", local.weights(x[1, ], x[2:4, ], 0.01), x[2:4, ]), nrow = 3)) -
    x[1, ]
## Error in solve.default(gram, rep(1, k)) :
##   system is computationally singular: reciprocal condition number = 1.04765e-17
## [1]  0.01091407 -0.06487090
```

The error message alerts us that the unregularized attempt to solve for the weights failed, since the determinant of the Gram matrix was as close to zero as makes no difference, hence it's uninvertible. (The error message could be suppressed by adding a `silent=TRUE` option to `try`; see `help(try)`.) However, with just a touch of regularization ($\alpha = 0.01$) we get quite reasonable accuracy.

Let's test our iterative solution. Pick $k = 2$, each row of the weight matrix should have two non-zero entries, which should sum to one. (We might expect some small deviation from 1 due to finite-precision arithmetic.) First, of course, the weights should match what the `local.weights` function says.

```
x.2NNs <- find.kNNs(x, 2)
x.2NNs[1, ]
## [1] 2 3
local.weights(x[1, ], x[x.2NNs[1, ], ], 0.01)
## [1]  1.9753018 -0.9753018
wts <- reconstruction.weights(x, x.2NNs, 0.01)
sum(wts[1, ] != 0)
## [1] 2
all(rowSums(wts != 0) == 2)
## [1] TRUE
all(rowSums(wts) == 1)
## [1] FALSE
summary(rowSums(wts))
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       1       1       1       1       1       1
```

Why does `summary` say that all the rows sum to 1, when directly testing that

```
local.weights.for.index <- function(focal, x, NNs, alpha) {
    n = nrow(x)
    stopifnot(n > 0, 0 < focal, focal <= n, nrow(NNs) == n)
    w = rep(0, n)
    neighbors = NNs[focal, ]
    wts = local.weights(x[focal, ], x[neighbors, ], alpha)
    w[neighbors] = wts
    return(w)
}
```

Code Example 44: *Finding the weights for the linear approximation of a point given its index, the data-frame, and the matrix of neighbors.*

says otherwise? Because some rows don't *quite* sum to 1, just closer-than-display tolerance to 1.

```
sum(wts[1, ]) == 1
## [1] FALSE
sum(wts[1, ])
## [1] 1
sum(wts[1, ]) - 1
## [1] -1.110223e-16
summary(rowSums(wts) - 1)
##       Min.    1st Qu.     Median       Mean    3rd Qu.       Max.
## -2.220e-16  0.000e+00  0.000e+00 -1.147e-17  0.000e+00  2.220e-16
```

So the constraint is satisfied to $\pm 2 \cdot 10^{-16}$, which is good enough for all practical purposes. It does, however, mean that we have to be careful about testing the constraint! Fortunately, the `all.equal()` function understands about numerical precision:

```
all.equal(rowSums(wts), rep(1, ncol(wts)))
## [1] TRUE
```

Of course, iteration is usually Not the Way We Do It in R — especially here, where there's no dependence between the rows of the weight matrix.[13] What makes this a bit tricky is that we need to combine information from two matrices — the data frame and the matrix giving the neighborhood of each point. We could try using something like `mapply` or `Map`, but it's cleaner to just write a function to do the calculation for each row (Example 44), and then apply it to the rows.

As always, check the new function:

```
w.1 = local.weights.for.index(1, x, x.2NNs, 0.01)
w.1[w.1 != 0]
## [1]  1.9753018 -0.9753018
which(w.1 != 0)
## [1] 2 3
```

[13] Remember what makes loops slow in R is that every time we change an object, we actually create a new copy with the modified values and then destroy the old one. If n is large, then the weight matrix, with n^2 entries, is very large, and we are wasting a lot of time creating and destroying big matrices to make small changes.

```
reconstruction.weights.2 <- function(x, neighbors, alpha) {
    n = nrow(x)
    w = sapply(1:n, local.weights.for.index, x = x, NNs = neighbors, alpha = alpha)
    w = t(w)
    return(w)
}
```

CODE EXAMPLE 45: *Non-iterative calculation of the weight matrix.*

```
coords.from.weights <- function(w, q, tol = 1e-07) {
    n = nrow(w)
    stopifnot(ncol(w) == n)
    stopifnot(all(abs(rowSums(w) - 1) < tol))
    M = t(diag(n) - w) %*% (diag(n) - w)
    soln = eigen(M)
    coords = soln$vectors[, ((n - q):(n - 1))]
    return(coords)
}
```

CODE EXAMPLE 46: *Getting manifold coordinates from approximation weights by finding eigenfunctions.*

So (at least for the first row!) it has the right values in the right positions.

Now the final function is simple (Example 45), and passes the check:

```
wts.2 = reconstruction.weights.2(x, x.2NNs, 0.01)
identical(wts.2, wts)
## [1] TRUE
```

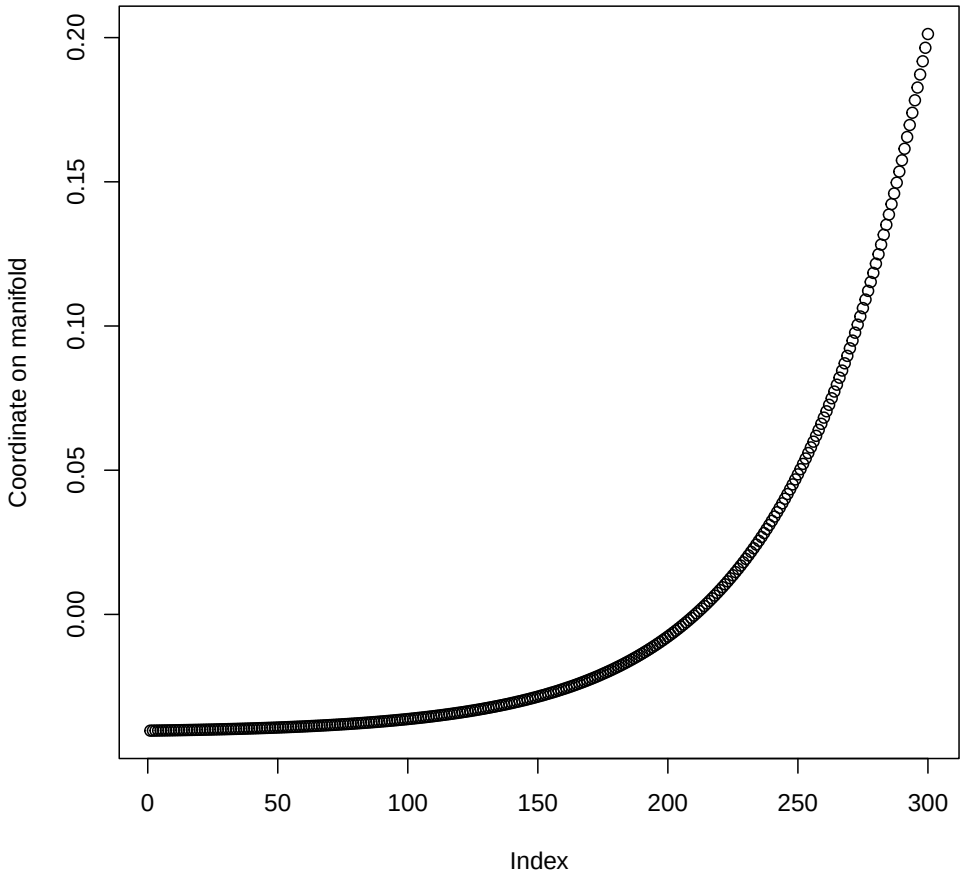
G.4.3 Calculating the Coordinates

Having gone through all the eigen-manipulation, this is a straightforward calculation (Example 46).

Notice that **w** will in general be a very **sparse** matrix — it has only k nonzero entries per row, and typically $k \ll n$. There are special techniques for rapidly solving eigenvalue problems for sparse matrices, which are not being used here — another way in which this is not an industrial-strength version.

Let's try this out: make the coordinate (with $q = 1$), plot it (Figure G.5), and check that it really is monotonically increasing, as the figure suggests.

```
spiral.lle = coords.from.weights(wts, 1)
plot(spiral.lle, ylab = "Coordinate on manifold")
```



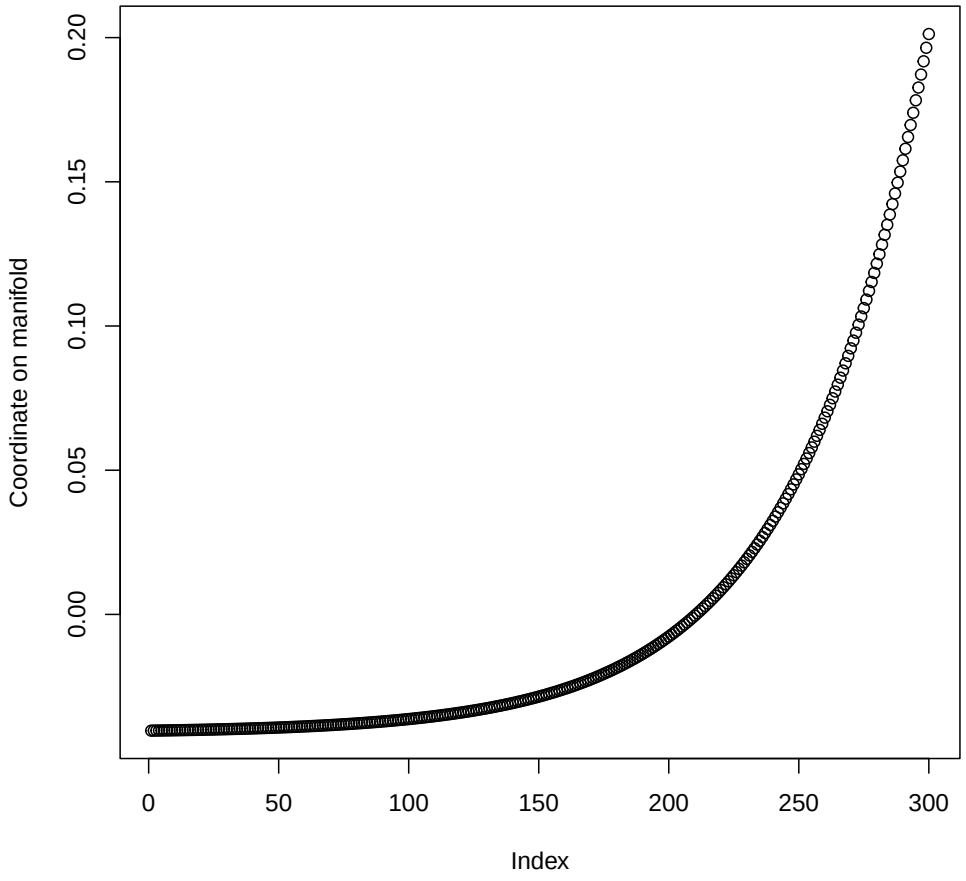

```
all(diff(spiral.lle) > 0)
## [1] TRUE
```

So the coordinate we got through LLE increases along the spiral, just as it should, and we have successfully recovered the underlying structure of the data. To verify this in a more visually pleasing way, Figure G.6 plots the original data again, but now with points colored so that their color in the rainbow corresponds to their inferred coordinate on the manifold.

Before celebrating our final victory, test that everything works when we put it together:

```
all.equal(lle(x, 1, 2), spiral.lle)
## [1] TRUE
```

□




```
plot(coords.from.weights(wts, 1), ylab = "Coordinate on manifold")
```

Figure G.5 Coordinate on the manifold estimated by locally-linear embedding for the spiral data. Notice that it increases monotonically along the spiral, as it should.

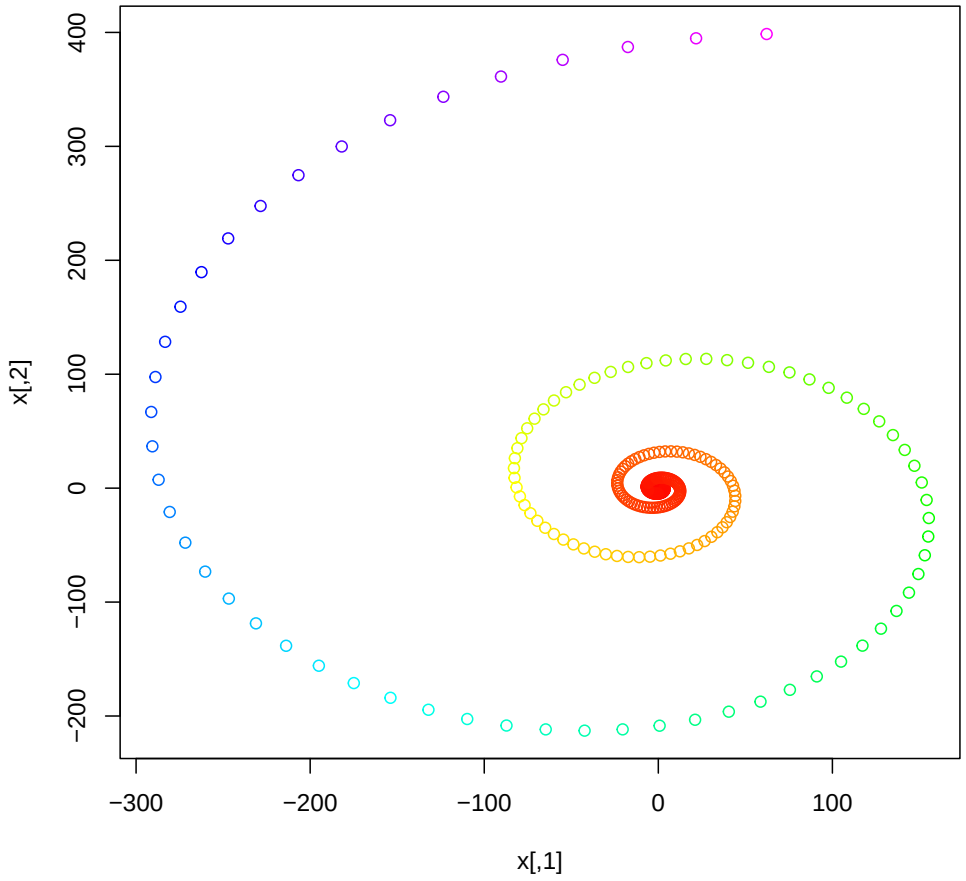
G.5 Further Reading

[[TODO: Write, insert refs.]]

[[SNE]]

[[Eigenmaps: Belkin and Niyogi (2003)]]]

[[Diffusion maps: form a similarity graph for the data, and then use as coordinates projections on the eigenvectors of the graph Laplacian Lee and Wasserman




```
plot(x, col = rainbow(300, end = 5/6)[cut(spiral.lle, 300, labels = FALSE)])
```

Figure G.6 The original spiral data, but with color advancing smoothly along the spectrum according to the intrinsic coordinate found by LLE.

(2010); distinct from LLE[14]]] [[Diffusion maps: see also `http://www.stat.cmu.edu/~cshalizi/350/lectures/15/lecture-15.pdf`]]

[[Manifold learning]]

[[LLE: note that, Like PCA, not really fitting a probability model (so pure

[14] In fact, in some cases, it can be shown (Belkin and Niyogi, 2003, §5) that the matrix in the LLE minimization problem is related to the Laplacian, because $(\mathbf{I}-\mathbf{w})^T(\mathbf{I}-\mathbf{w}) \approx \frac{1}{2}\mathbf{L}^2$. Since the powers of $\mathbf{L}$ have the same eigenvectors as $\mathbf{L}$, when this holds the coordinates we get from the diffusion map are *approximately* the same as the LLE coordinates.

data analysis, rather than statistical inference) — refs. on ideas more like factor models, with an inferential/probabilistic component]]

Exercises

G.1 Let $\mathbf{b}$ be any $n \times n$ matrix. Show that if $\vec{v}$ is an eigenvector of $\mathbf{b}$, then it is also an eigenvector of $\mathbf{b}^2$, and of any power of $\mathbf{b}$. Conclude that $\mathbf{b}$ and $\mathbf{b}^2$ have the *same* eigenvectors. (*Hint:* how many eigenvectors does each matrix have?) What happens to the eigenvalues?

G.2 Find the matrix $\mathbf{A}$ which expresses the mean-zero constraint in the form $\mathbf{AY} = 0$.

G.3 In local linear embedding, we obtain an $n \times n$ matrix $\mathbf{w}$, where w_{ij} is the weight on $\vec{x}_j$ we use to reconstruct $\vec{x}_i$. Each row of $\mathbf{w}$ sums to one. We then try to find coordinates $y_1, y_2, \ldots y_n$ which minimize

$$\Phi(\mathbf{Y}) = \sum_{i=1}^{n} \left(y_i - \sum_{j=1}^{n} w_{ij} y_j \right)^2 \tag{G.42}$$

where $\mathbf{Y}$ is the $n \times 1$ matrix of y_i values (this is the $q = 1$ case, for simplicity). Above, in Eq. G.34, we showed that this is the same as minimizing

$$\Phi(\mathbf{Y}) = \mathbf{Y}^T \mathbf{M} \mathbf{Y} \tag{G.43}$$

where

$$\mathbf{M} = ((\mathbf{I} - \mathbf{w})^T (\mathbf{I} - \mathbf{w})) \tag{G.44}$$

1. Show that $\mathbf{M}$ is a symmetric matrix.
2. Show that $\mathbf{1}$ is an eigenvector of $\mathbf{M}$, and that its eigenvalue is zero.
3. Show that $\Phi(\mathbf{Y}) = \Phi(\mathbf{Y} + c\mathbf{1})$, where c is any constant and $\mathbf{1}$ is the $n \times 1$ matrix whose entries are all 1s. (*Hint:* one way is to use the previous two parts.)
4. Show that $\Phi(\mathbf{Y})$ is minimized by $\mathbf{Y} = 0$.
5. To avoid the trivial solution of setting all the y_i to zero, we impose the constraint that $n^{-1} \sum_{i=1}^{n} y_i^2 = 1$. We use a Lagrange multiplier to enforce this constraint; write down the Lagrangian for the constrained minimization problem.
6. Show that a solution $\mathbf{Y}$ to the constrained minimization problem must be an eigenvector of $\mathbf{M}$.

Appendix H

Rudimentary Graph Theory

[[TODO: Streamline, and re-integrate into the graphical models chapter]]

A **graph** G is built out of a set of **nodes** or **vertices**, and **edges** or **links** connecting them. The edges can either be directed or undirected. A graph with undirected edges, or an undirected graph, represents a symmetric binary relation among the nodes. For instance, in a social network, the nodes might be people, and the relationship might be "spends time with". A graph with directed edges, or **arrows**, is called a directed graph or **digraph**[1], and represents an asymmetric relation among the nodes. To continue the social example, the arrows might mean "admires", pointing from the admirer to the object of admiration. If the relationship is reciprocal, that is indicated by drawing a *pair* of arrows between the nodes, one in each direction (as between A and B in Figure H.1).

A **directed path** from node V_1 to node V_2 is a sequence of edges, beginning at V_1 and ending at V_2, which is connected and which follows the orientation of the edges at each step. An **undirected path** is a sequence of connected edges ignoring orientation. (Every path in an undirected graph is undirected.) If there is a directed path from V_1 to V_2 and from V_2 to V_1, then those two nodes are **strongly connected**. (In Figure H.1, A and C are strongly connected, but A and D are not.) If there are undirected paths in both directions, they are **weakly connected**. (A and D are weakly connected.) Strong connection implies weak connection (Exercise 1). We also stipulate that every node is strongly connected to itself.

Strong connection is an equivalence relation, i.e., it is reflective, symmetric and transitive (Exercise 2). Weak connection is also an equivalence relation (Exercise 3). Therefore, a graph can be divided into non-overlapping **strongly connected components**, consisting of maximal sets of nodes which are all strongly connected to each other. (In Figure H.1, A, B and C form one strongly connected component, and D and E form components with just one node; Exercise **??**.) It can also be divided into **weakly connected components**, maximal sets of nodes which are all weakly connected to each other. (There is only one weakly connected component in the graph. If either of the edges into D were removed, there would be two weakly connected components; Exercise **??**.)

A **cycle** is a directed path from a node to itself. The existence of two distinct nodes which are strongly connected to each other implies the existence of a cycle,

[1] Or, more rarely, a **Guthrie diagram**.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

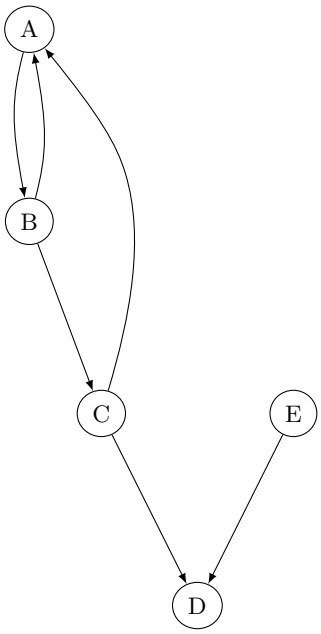


Figure H.1 Example for illustrating the concepts of graph theory.

and vice versa (Exercise 6). A directed graph without cycles is called **acyclic**. Said another way, an acyclic graph is one where all the strongly connected components consist of individual nodes. The weakly connected components can however contain an unlimited number of nodes.

In a **directed acyclic graph**, or **DAG**, it is common to refer to the nodes connected by an edge as "parent" and "child" (so that the arrow runs from the parent to the child). If there is a directed path (of any length) from V_1 to V_2, then V_1 is the **ancestor** of V_2, which is the **descendant** of V_1. In the jargon, the ancestor/descendant relation is the **transitive closure** of the parent/child relation.

H.1 Exercises

1. Prove that if two nodes are strongly connected, they are also weakly connected. Draw a graph in which two nodes are weakly connected but not strongly connected.
2. Prove that strong connection between nodes is an equivalence relation.
    1. *Reflexive* Prove that every node is strongly connected to itself.
    2. *Symmetric* Prove that if A is strongly connected to B, then B is strongly connected to A.
    3. *Transitive* Prove that if A is strongly connected to B, and B is strongly connected to C, then A is strongly connected to C.

3. Prove that weak connection between nodes is an equivalence relation. Divide the proof into parts as in Exercise 2.
4. Verify that the graph in Figure H.1 has three strongly connected components, $\{A, B, C\}$, $\{D\}$ and $\{E\}$.
5. Verify that the graph in Figure H.1 has only one weakly connected component. Check that if either of the edges into D were removed, there would be two weakly connected components — what would they be?
6. Prove that if A is strongly connected to even one other node $B \neq A$, then there is a cycle in the graph.

Appendix I

Missing Data

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

There was a point to the blank page, beyond the obvious joke. Tautologously, missing data is data we do not have. We don't know what it would have been. Anything we say about it is guesswork, based on assumptions. *All* statistical inference rests on assumptions, but they are especially hard to take for granted, and especially hard to check or to justify, when they're assumptions about things we *know* we don't have evidence about.

To be a bit more formal, "missing data" refers to situations where we are able to record some variable for some but not all of the units of analysis in our data set. Conventionally, in R, those are `NA` values in our data frame[1] Variables we never measure are not **missing**, but **latent** or **hidden**, or indeed **ignored**. If we do a face-to-face survey about people's finances, our survey-takers would typically be able to record the interviewees' eye colors and the number of times they say "um", but wouldn't do so: those variables are ignored. If some but not all of those surveyed refuse to say how much they spend on housing every month, *that* is missing data. Missing data is very, very common in the real world.

We would (in general) draw different inferences if only we had the **complete** data, the data we actually got plus the data that is, in fact, missing. But (to hammer home the point), precisely because it's missing, we don't have direct evidence about *how* our inferences would change. Any effort to take missing data into account will rely on assumptions about what we would have seen. Any analysis which tries to just ignore missing-ness is only going to make sense under assumptions of its own.

To give an initial feel for the kinds of problems which arise, start with Figure I.1[2]. It's a scatter-plot of a kind you've seen many times already, where you may suppose we're interested in how Y depends on X.

Looking carefully at Figure I.1, you may notice that there are more tick-marks on the horizontal axis than there are points in the scatter-plot. These tick-marks come from data points where I am treating Y as missing but X as observed. (In fact, to keep things simple, I have made it so that X is *never* missing.) In the observed data, higher values of X clearly predict higher values of Y. Whether that is also true over-all depends on what the missing values are like. Figure I.2 illustrates a few different possibilities, all of which are equally compatible, logically, with the observations in Figure I.1.

More concretely, Figure I.2 illustrates four *different* data-generating processes, filling in the missing values. The relationship between X and Y implied by the filled circles in Figure I.2b is clearly very different from the one implied by the

[1] R, as a modern computer language, is capable of doing arithmetic with NA values, and (correctly) "'propagates" them, so that anything plus `NA` is also `NA`, etc. In earlier times, however, many computing systems lacked an `NA` value, and so particular numbers were sometimes used to "code" missing values. Common choices included -1, 0, and (very insidiously) 99 and 999. (For a case where 99 was the missing value code, but some 99s were apparently mis-entered as 88s, leading to very surprising conclusions, see Kahn and Udry (1986).) I myself once worked with a data-set which coded missing values of worker's ages 66, because everyone was supposed to retire at 65.

[2] This is a simulation, not real data, but it's inspired by a project I participated in on predicting which people who had been arrested could be safely released while awaiting trial. The code generating it appears on p. 759, but please don't peek ahead.

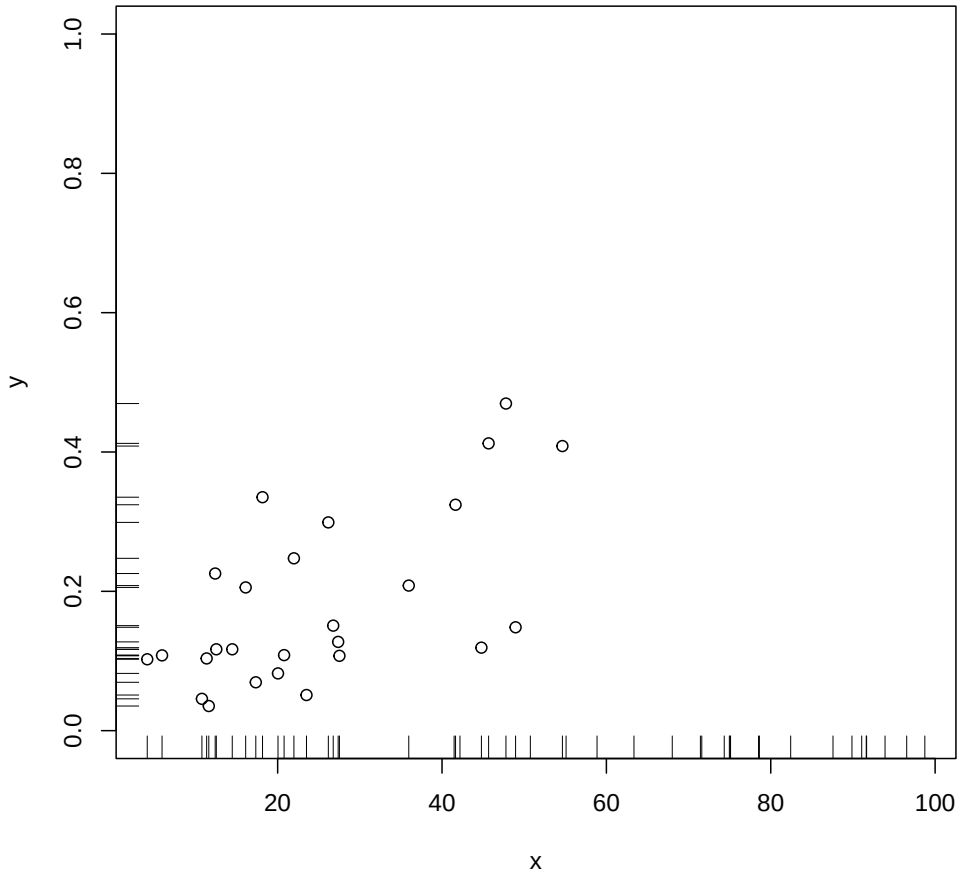


Figure I.1 Running example for illustrating missing-data issues. The rug-plots along the axes indicate the marginal distributions of the observed (not missing) values. The generating code is deliberately hidden here, but will be given at the end of the chapter.

empty diamonds in Figure I.2d. But all four processes, and infinitely many others, are equally compatible with the fully-observed data points. This makes it vivid that there are two big tasks when dealing with missing data:

1. Almost all of our computational procedures presume complete data sets. How, as a technical matter, can we use data sets with missing values in such procedures, and under which assumptions are different techniques good ideas?
2. How, if it all, can we check assumptions about missing-ness?

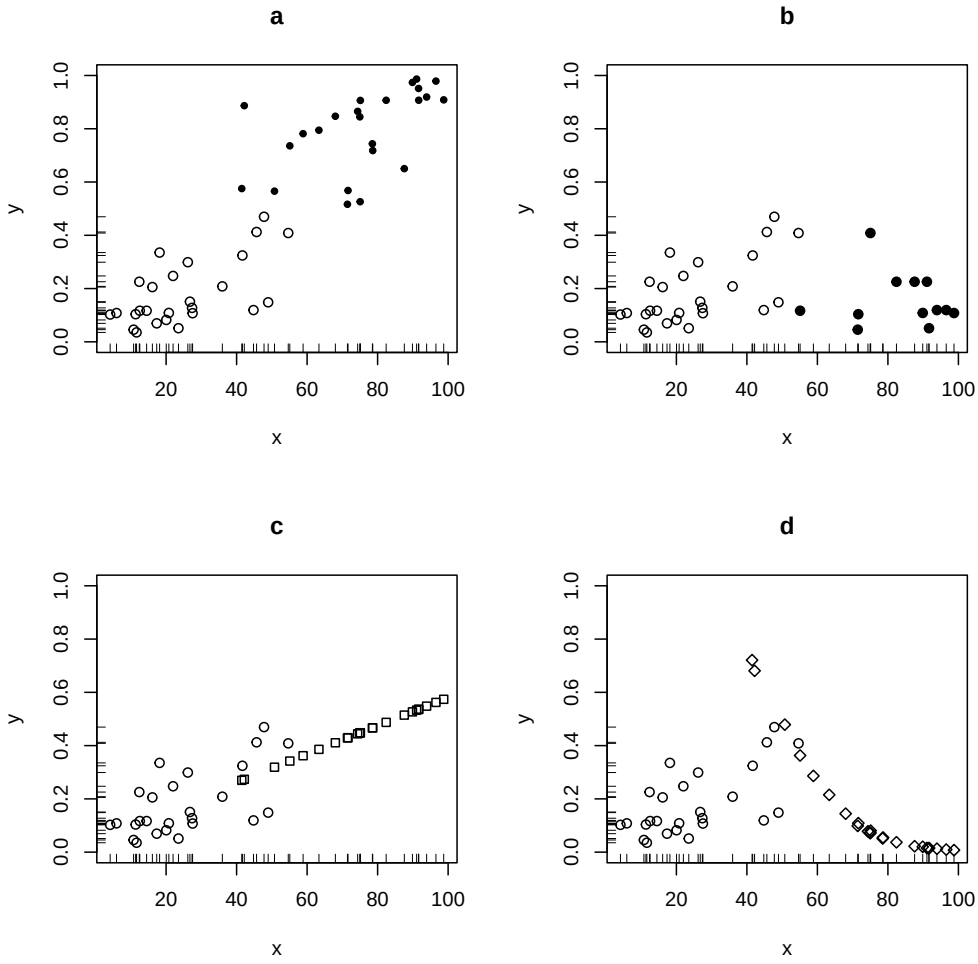


Figure I.2 Some possible complete data sets, all compatible with the observations in Figure I.1. The four point shapes indicate four distinct data-generating processes. In all four, the hollow circles are the fully-observed data points.

Since the right techniques for handling missing data depends on assumptions, you would be forgiven for thinking that assumption-checking is more important than techniques. Unfortunately, as we'll see in §I.4, there are good reasons why statisticians have given much more attention to techniques.

I.0.1 Notation and Preliminaries

We will often need a concise way to refer to whether or not a variable is missing. M_Y will be the indicator variable which is 1 when Y is missing and 0 when Y is observed. If we need to refer to its value in a particular observation i, we'll write that M_{Y_i}. We will also often refer to the probability that an observation is *not* missing, $\Pr(M_Y = 0)$, the **inclusion** or **capture** probability[3], $\pi = 1 - \mathbb{E}[M_Y]$. When this is conditioned on $X = x$, we'll write $\pi(x)$. When we need to collectively refer to all the fully-observed values of Y, they will be Y_{obs} (realization, y_{obs}), and, likewise, the collection of all missing values will be Y_{miss} (realization, y_{miss}).

We would like to know about the complete-data distribution, what generated the data before some of it went missing. Depending on how ambitious we are, this might be $\Pr(Y = y)$, or $\Pr(Y = y|X = x)$, or $\Pr(Y = y, X = x)$. (As usual, everything works out just the same for continuous probability densities as for discrete probability mass functions, so I'll just write things out for the latter.) Basic probability tells us that

$$
\begin{aligned}
&\Pr(Y = y, X = x) \\
&= \Pr(Y = y, X = x, M_Y = 0) + \Pr(Y = y, X = x, M_Y = 0) && \text{(I.1)} \\
&= \Pr(Y = y, X = x|M_Y = 0)\Pr(M_Y = 0) + \Pr(Y = y, X = x|M_Y = 1)\Pr(M_Y = && \text{(I.2)} \\
&= \Pr(Y = y, X = x|M_Y = 0)\,\pi + \Pr(Y = y, X = x|M_Y = 1)\,(1 - \pi)\ , && \text{(I.3)}
\end{aligned}
$$

that

$$
\begin{aligned}
&\Pr(Y = y|X = x) \\
&= \Pr(Y = y, M_Y = 0|X = x) + \Pr(Y = y, M_Y = 1|X = x) && \text{(I.4} \\
&= \Pr(Y = y|X = x, M_Y = 0)\Pr(M_Y = 0|X = x) + \Pr(Y = y|X = x, M_Y = 1)\Pr(M_Y = 1|X = && \text{(I.5)} \\
&= \Pr(Y = y|X = x, M_Y = 0)\,\pi(x) + \Pr(Y = y|X = x, M_Y = 1)\,(1 - \pi(x))\ , && \text{(I.6}
\end{aligned}
$$

and that

$$
\begin{aligned}
\Pr(Y = y) &= \Pr(Y = y, M_Y = 0) + \Pr(Y = y, M_Y = 1) && \text{(I.7)} \\
&= \Pr(Y = y|M_Y = 0)\,\pi + \Pr(Y = y|M_Y = 1)\,(1 - \pi)\ . && \text{(I.8)}
\end{aligned}
$$

In every case, what we want is the complete-data expression on the left-hand side. But what we might be able to identify from the data are just the first parts of the sums on the right, where we're conditioning on $M_Y = 0$, i.e., conditioning on Y being observed. The complete-data expressions will be identified from observations only under assumptions which somehow tie the missing-data terms, where $M_Y = 1$, to things we observe.

[3] It may seem perverse to have a *missing* indicator and an *inclusion* probability, but these choices will simplify formulae later, and anyway are conventional.

I.1 Deletion, and Missing-at-Random Assumptions

The simplest way to handle missing data is to not use incomplete records — to **delete** them[4]. This can be done in multiple ways, depending on how aggressive one wants to be about using the variables that aren't missing.

One extreme of deletion is to drop all records which are incomplete in any variable[5] This has come to be called **listwise deletion**. Its greatest advantage is stark simplicity. With a data-frame in R, this amounts to taking the columns you need, and then dropping any row with an `NA`, as performed by `na.omit()`. This is the default behavior of most model-fitting functions, once they've determined which columns of the data-frame they will actually use.

Just because something is R's default behavior does not, however, mean it is a good idea[6]. While no *good* might come of throwing away some of our data at random, random deletion at least wouldn't lead to *systematic* mistakes. But the one thing we know about the rows with missing values is that they are systematically different from the complete rows in an important way — namely, some variables are missing! Listwise deletion would seem, on the face of it, to be a recipe for creating biased samples and introducing systematic errors. If it's to make any statistical sense, very strong assumptions will be required.

I.1.1 Practicalities of Data Analysis After Deletion

If we make such assumptions, though, our life can be very straightforward. Figure I.3 shows what our running example data set looks like after deletion, with a simple smoothing spline run through it.

The only even slightly subtle thing to remember is that our sample size has shrunk: it's not the original number of data points, but just the number of fully-observed data points.

I.1.2 Assumptions Justifying Deletion

I.1.2.1 Missing Completely at Random

One strong assumption which would justify deletion is that of "missingness completely at random" (MCAR), that $M_Y \perp\!\!\!\perp Y, X$. The idea is that, in effect, Somebody went down the rows of the data frame, tossed a coin for each row, and erased Y whenever the coin came up heads. In a situation like this, the complete rows really are a representative sample of the complete data. Formally, the independence assumed by MCAR means that

$$\Pr(Y = y, X = x) = \Pr(Y = y, X = x | M_Y = 0) \tag{I.9}$$

[4] Despite the name, it is generally a bad idea to actually delete them from your main data file! Instead, drop rows from working copies of the data-frame in your code.

[5] That is, any variable you are actually using in a given analysis.

[6] By now, you've learned this lesson when it comes to the precision with which numerical results should be reported.

So anything we could identify from the complete data (because it's a function of the joint distribution) is also something we can identify from the distribution of the data after deletion. (Can you prove this, using Eq. I.3?) Similarly, since our estimates are built using an effectively-random subset of the total sample, it's as though we just had a somewhat smaller random sample to begin with, or at least one that's no more biased than the complete data. Anything procedure which should be consistent with the complete data should also be consistent after deletion, *if we assume MCAR.*

MCAR is, obviously, a very strong assumption, and, honestly, hard to believe in lots of real-world situations. It's not logically impossible that the people who won't tell interviewers how much they spend on housing are *just like* those who do say, both in terms of their income and their housing expenses, but it's not very plausible. In fact, because $M_Y \perp\!\!\!\perp X, Y$ implies $M_Y \perp\!\!\!\perp X$, this is one of the easiest assumptions about missing-ness to *dis-prove* (see §I.4). In the case of Figure I.1, it's clear by inspection that large and small values of X predict very different rates of missing-ness for Y, so MCAR is very implausible here[7].

I.1.2.2 Missing at Random

Because MCAR is so strong, and usually so implausible, but deletion is so tempting, people have looked for weaker assumptions which would still justify ignoring incomplete records. One favorite has come to be called **missing-at-random** (MAR), or **ignorable missingness** or **uninformative missingness**. It is simply that

$$M_Y \perp\!\!\!\perp Y|X \tag{I.10}$$

By the definition of conditional independence, this is equivalent to saying that, for all x, y,

$$\Pr(Y = y|X = x, M_Y = 0) = \Pr(Y = y|X = x, M_Y = 1) \tag{I.11}$$

In words: *given* X, the missing values of Y follow exactly the same probability distribution as the observed values[8]. From this, and Eq. I.6, it follows that

$$\Pr(Y = y|X = x) = \Pr(Y = y|X = x, M_Y = 0) \tag{I.12}$$

Thus, under MAR, any function of the complete-data conditional distribution can be calculated directly from the observed conditional distribution. In fact, the complete-data *joint* distribution is identified under MAR, though it's not equal to the observed joint distribution (Exercise 1).

Because MAR is all about conditional independence, it's very natural to want to apply the graphical modeling ideas introduced in Chapter 18. Thus Figure I.4 shows the simplest, though by no means the only, graphical model in which MAR holds, but MCAR (in general) does not.

[7] How might you conduct a formal test, if you're not willing to believe me? *Hint:* Can you apply Chapter 9?

[8] The probability of being missing can change with X, so the marginal distributions $Y|M_Y = 0$ and $Y|M_Y = 1$ need not be equal; see Exercise 2.

It's natural to wonder if anything weaker than MAR could still justify deletion. The answer is pretty much "no". If we want Eq. I.12 to hold, so that the complete-data and observed conditional distributions are equal, then the assumption-free Eq. I.6 implies Eq. I.11, and we're back to MAR.

I.1.3 Partial Deletion

The disadvantage of listwise deletion is that records which are missing one variable might still have perfectly good values for other variables. If we assume missingness is uninformative, not using those records won't introduce biases, but it will reduce the precision of some statistics. In **partial deletion** (a.k.a. **pairwise deletion**, or **"available pairs"**), we use incomplete data points to calculate whatever statistics they can help with. For example, in linear regression, a point which is missing Y can still contribute to calculating $\mathbf{x}^T\mathbf{x}$. In fact, a point which also missing some columns of x could still contribute to *some* entries in $\mathbf{x}^T\mathbf{x}$. Thus, in partial deletion, we try to stretch out the data as much as possible. The ability to do this why I do *not* recommend just running `na.omit` on your data, even if you're willing to assume MAR (see also Exercises 1 and 2).

As usual, these advantages come at a price. Because different calculations are done with more-or-less different data sizes, it is *possible* to get mutually inconsistent results. Thus if you estimate $\mathbb{V}[X]$, $\mathbb{V}[Y]$ and $\mathrm{Cov}[X, Y]$ when Y is missing using partial deletion, you might get a correlation coefficient bigger than 1 or smaller than -1.

I.2 Informative Missingness, or Missing-Not-At-Random

The opposite of missing-at-random is, naturally, **missing-not-at-random**, or **MNAR**:

$$M_Y \not\perp Y|X \tag{I.13}$$

This also has the more comprehensible names of **non-ignorable missingness** or **informative missingness** (see Exercise 3).

When missingness is informative, arguments parallel to the ones we used to justify deletion under MAR tell us that deletion is a bad idea. Specifically,

$$\Pr(Y = y|X = x) \neq \Pr(Y = y|X = x, M_Y = 0) \tag{I.14}$$

so the observed data *doesn't* follow the same conditional distribution as the complete data, and deletion will give us a biased, systematically-distorted idea of that conditional distribution.

There is very little more which can be said at this level of generality at MNAR. In particular, while missingness is informative, it might or might not be *very* informative. It can also take many different forms. Two of the most common are **censoring** and **selection**.

I.2.1 Censoring

When Y values in certain ranges are just not observed, Y is **censored**. The most common examples of this are **right censoring**, when we never see $Y > y_{\max}$, and **left censoring**, when we never see $Y < y_{\min}$. The classic setting for right-censored data is when Y is the time to some event — how long patients live, or how long a machine lasts before breakdown, etc. Any patient who is still alive at the end of the study that collected the data *will* die eventually, so they have some survival time, but it's missing, and the missingness is directly a function of how long they've lived. As a classical problem, it has a classical solution, the "product-limit" or Kaplan and Meier (1958) estimator (implemented in, e.g., the `survival` package (Therneau, 2015)). This, however, rests on the assumption that the time at which each observation is censored is deterministic, or at least independent of the actual lifespan[9]. Lifespans or durations can also be left-censored, if events which happen too quickly don't show up in our data[10].

I.2.2 Selection

In many situations, we only get to observe Y for individuals (cases, etc.), which are somehow **selected** into one condition or another, but the process of selection is itself (supposed to be) sensitive to what Y *would be*. That's a rather vague, abstract statement, but concrete cases are very common. Admissions processes at many schools deliberately try to select students who will do well at the school, but we only get to see academic outcomes for students who were selected. If Y is any measure of academic outcomes, then, there's potentially informative missingness.

Ignoring such selection can be seriously misleading. As a little simulation to prove the point, I first generate test scores uniformly distributed between 200 and 1600, and then a subsequent grade in the range of 0 to 4, which is a linear and increasing, but noisy, function of the test scores:

```
n <- 1000
test.scores <- runif(n, min=200, max=1600)
gpas <- 4*(test.scores-200)/1400 + rnorm(n, sd=0.5)
```

The correlation between scores and outcomes is unsurprisingly high, 0.91. But if I only look at those who scored above, say, 1300, the correlation drops immensely, to 0.46. People who dislike using test-scores in admissions decisions sometimes point to evidence that such scores are poor predictors of success in academic programs among those admitted, which is often true, but is *exactly* what we would expect if such scores were *good* predictors and used for selection[11]

This is not just a literally-academic issue. Perhaps the most consequential place

[9] That is, it's assumed that the real life-time of unit i is Y_i, that it is lost to observation at a time L_i, that we see Y_i if $Y_i \leq L_i$, and that L_i is either constant for all i, or that $Y_i \perp\!\!\!\perp L_i$.

[10] This can be an issue when studying the lifespans of social or political movements, online fads, etc.

[11] This is a very old point — see Dawes (1975) — but still a valid one. (Dawes raises the additional, more subtle point that when those admitted are selected on multiple variables, say X_1 and X_2, we are conditioning on a collider (see ch. 19), which creates negative correlations among the predictors, and tends to make each predictor only weakly related to the outcome Y.) — I say all this as

such issues arise is in courts and prisons, which make decisions about who will be arrested, who will be released on bail between arrest and trial, who will be released on parole, etc. Increasingly, such decisions are made using predictive statistical models[12]. In developing these models, there is information about whether (for example) those released on bail show up for their trial, commit another crime[13], etc., but this information is missing for those who are kept in jail while awaiting trial. Any Y which measures those outcomes is potentially subject to informative missingness[14]

In these situations, it's *imaginable* that Y is really missing at random, conditional on the right set of X variables. After all, the people currently making decisions about bail (mostly judges) can't see the future to know what Y would be, but rather have to rely on cues and signals that actually exist when they make their decisions[15]. One can imagine bundling all those cues and signals up into an X variable, and then it would be positively unreasonable to think anything other than $Y \perp\!\!\!\perp M_Y|X$ (see Figure I.5). But if we as *data-analysts* don't have access to exactly the X used by the decision-makers, but instead some other variable X', it could very easily be that $Y \not\perp\!\!\!\perp M_Y|X'$, so that missingness is informative *for us* (see Figure I.5 again).

In efforts to develop models for deciding on pre-trial release, for example, one commonly has data about the demographic characteristics of the arrestee (age, sex, etc.), what they have been arrested for, and their prior history of arrests and convictions[16]. But a judge might consider other factors, such as the reputation of the arresting officer, or testimony about the character of the arrestee, or the arrestee's appearance[17] The decisions made by the selectors would then reflect some information about Y which is not available to us in X'. This can, in fact, completely change the apparent implications of the variables in X'. If, overall, people who have been people who have been convicted of many violent crimes are especially likely to re-offend before trial if granted bail, they might

someone who generally *doesn't* like emphasizing standardized tests in admissions decisions, but for other reasons.

[12] I have been myself involved in an effort to evaluate pre-trial release models for a non-profit organization.

[13] More precisely: about whether they are *arrested* for another crime.

[14] The same issue arises for credit risk: lenders try to select loan applicants who will repay, but we don't see whether those denied loans would have repaid. Any Y reflecting repayment is, then, potentially subject to informative missingness. But let's stick with crime and punishment, rather than banking, for right now.

[15] This is why econometricians sometimes refer to MAR as "selection on observables". It is also one way selection differs, conceptually, from censoring, where M_Y depends directly on Y.

[16] Though not always. The prior legal history is itself missing more often than anyone should like. In large part this is because different organizations (e.g., police vs. courts vs. prisons vs. parole offices), even within the same legal jurisdiction, do a very bad job at sharing and linking up their records. Even the same organization may not have a good way to keep track of whether the Joe Smith now on trial for theft is the same as the J. E. Smith previously convicted of fraud.

[17] As this last indicates, nothing says that everything in X' has to be either ethically legitimate or rationally linked to Y. (Devising situations where it is legitimate, rational and legal for a judge to be influenced by how an arrestee looks when deciding whether or not to grant bail is left as an exercise for your ingenuity.)

appear unusually *safe*, because judges only grant them bail when (say) there is abundant and credible testimony that they have reformed. That is, when these decisions are made accurately, using cues and signs that really are informative, $\Pr(\text{re-offend}|\text{history of violence}, M_{=}0)$ can be very low, even though $Prob\text{re-offend}|\text{history of violence}$ is very high. Under less extreme circumstances, a history of violent crime might seem to pose less of a risk than a history of non-violent crime, for the same reason.

Censoring and selection are not the only two kinds of informative missingness, but it's almost impossible to give a complete catalog. In every event, for further work, it's important to investigate the precise *mechanisms* leading to missingness. This needs to be done at two levels, the statistical or probabilistic, and the substantive.

I.2.3 The "Missingness Mechanism", Statistically Considered

In the statistics of missing data, the "missingness mechanism" has come to refer to the conditional distribution

$$\Pr(M_Y = 1|X = x, Y = y) = 1 - \pi(x, y) \tag{I.15}$$

(Of course, it works just as well to know the conditional probability that $M_Y = 0$.) Recall that when we're trying to find the conditional distribution of Y given X, the data lets us identify

$$\Pr(Y = y|X = x, M_Y = 0)\Pr(M_Y = 0|X = x) \tag{I.16}$$

but (Eq. I.6) we need that *plus*

$$\Pr(Y = y|X = x, M_Y = 1)(1 - \pi(x)) \tag{I.17}$$

to get $\Pr(Y = y|X = x)$. But basic probability (Exercise 5) tells us that

$$\Pr(Y = y|M_Y = 1, X = x) = \frac{\Pr(M_Y = 1|X = x, Y = y)\Pr(Y = y|X = x)}{\Pr(M_Y = 1|X = x)} \tag{I.18}$$

so (Exercise 6)

$$\Pr(Y = y|X = x) = \Pr(Y = y|X = x, M_Y = 0)\frac{\pi(x)}{\pi(x, y)} \tag{I.19}$$

so long as $\pi(x, y) > 0$. We thus have an expression for the complete-data conditional distribution, in terms of the observable conditional distribution, and the missingness mechanism, or the inclusion probabilities. Knowing the latter allows us to, so to speak, undo the distortions due to missingness. And, because we've assumed nothing but basic probability to get Eq. I.19, *any* assumption which is strong enough to let us identify $\Pr(Y|X)$ under MNAR has to either *be* an assumption about $\pi(x)/\pi(x, y)$, or has to *imply* the form of that ratio[18].

[18] In fact, MAR is the assumption that $\pi(x)/\pi(x, y) = 1$ for all x, y, so that the exact inclusion probabilities can be ignored (Exercise 7).

I.2.3.1 Example: Heckman Selectivity[19]

An example of modeling assumptions which define a missingness mechanism comes from a situation of selection that originally arose in studying the distribution of wages. A basic model of wage labor is that every individual has a "reservation wage", an amount that they would have to be paid to accept employment at all[20]. People may refuse offers of employment above their reservation wage, but they will definitely refuse offers below their reservation wage. Some people may therefore not be employees (at any given time) because they haven't received any offer exceeding their reservation wage. If we want to know how the wages people can command in the labor-market vary with their characteristics, it's important to have some idea of what those missing wages would have been.

Stated at this level of generality, the problem is unsolvable. The distribution of wages below the reservation wage could, logically speaking, be absolutely anything, without altering the observable distribution at all. Economic theory does not, in this case, provide any real constraints either. As they have so often done when faced with an unsolvable problem and unguided by substantive economics, econometricians have responded by trying to make linear regression work. Specifically, in the Heckman (1976) model, we assume that (log) wages Y are linear in a covariate X,

$$Y = \beta_1 x + \epsilon \tag{I.20}$$

as are (log) reservation wages,

$$R = \beta_2 x + \eta \tag{I.21}$$

with ϵ and η being independent of X and sharing a Gaussian distribution with mean 0 and variance matrix $\mathbf{\Sigma}$. We further assume that we see the wages of person i if, and only if, $Y_i > R_i$,

$$M_Y = \mathbb{1}(Y < R) \tag{I.22}$$

Under these assumptions, you can show that

$$\pi(x) = \Phi(x(\beta_1 - \beta_2)/\sigma) \tag{I.23}$$

where Φ is the standard Gaussian CDF, and σ^2 is the variance of $\eta - \epsilon$ (Exercise 82). Since the inclusion probabilities are observable, the composite parameter $(\beta_1 - \beta_2)/\sigma$ can be identified from observations (under all these assumptions). A similar but longer calculation (Exercise 84) shows that

$$\mathbb{E}\left[Y|X = x, M_Y = 0\right] = \beta_1 x - c\frac{\phi(x(\beta_1 - \beta_2)/\sigma)}{\Phi(x(\beta_1 - \beta_2)/\sigma)} \tag{I.24}$$

where ϕ is the standard Gaussian PDF, and c is the covariance between ϵ and

[19] My treatment of this classic topic in econometrics is heavily indebted to Manski (2007, §§2.6 and 4.1–4.2).

[20] As opposed to continuing to look for work in hopes of better jobs, going into business for themselves, dealing with family responsibilities, sleeping under bridges and stealing bread, or whatever.

$(\eta - \epsilon)/\sigma$. Once we know $(\beta_1 - \beta_2)/\sigma$, this lets us identify β_1, which is usually what's of interest.

There are several points worth making about this example.

1. You can extend the same logic to many situations of selection beyond wages. For instance, if we're studying life-spans, we can have Y_i be the life of unit i, and R_i be the amount of time between when unit i appears and the end of the study period. Then we only get to observe a completed lifespan if $Y_i \geq R_i$. It is thus unsurprising that Heckman (1976) has been cited over 5500 times[21].
2. Extending the model to multiple predictor variables is straightforward, if tedious, provided both Y and R remain linear in the coordinates of X.
3. We can identify β_1 without ever having to record the reservation wage R. This is handy, because usually we can't measure R at all.
4. The identifying power of the assumptions breaks down if we *don't* assume that $(\epsilon, \eta) \perp\!\!\!\perp X$ — for instance, heteroskedasticity in ϵ or η is in general enough to make things unidentifiable again.
5. Economic theory provides *absolutely no reason* to think that (log) wages and (log) reservation wages should be jointly Gaussian, with conditional means which are linear in the observed features, and homoskedastic disturbances. Many of those thousands of citations come from other areas of social science, like sociology and education, where theory is even less definite, and no more supportive of these assumptions.
6. We can use data to check whether Eq. I.23 holds, because it concerns purely observable quantities, with one unknown parameter. (In terms of the model, that's a composite, $\frac{\beta_1 - \beta_2}{\sigma}$, but that's still only one adjustable parameter.) We can also use data to check whether Eq. I.24 holds, since *that's* a parameteric form with three unknown parameters (β_1, c and $\frac{\beta_1 - \beta_2}{\sigma}$), albeit one which is non-linear in x. One could even, in principle, test whether the data imply the *same* value for $\frac{\beta_1 - \beta_2}{\sigma}$ in the two equations. But (and this is crucial) *we cannot test the full model*. There are many different models which would *also* imply both Eqns. I.23 and I.24. (The simplest of these, though perhaps not the most plausible, would say that inclusion probabilities follow Eq. I.23, while $Y \perp\!\!\!\perp M_Y | Z$, and $\mathbb{E}[Y|X = x]$ also follows the form given by Eq. I.24.)

I.2.4 Mechanisms of Missingness, Substantively

I have said that the "missingness mechanism", in the jargon, is just the function $\pi(x, y)$ (or its complement). Mathematically, if we know this, we know how to deal with our missing data. This immediately raises the question of *how we might learn it*. The only real way is to study how and why, exactly, some, but only some, of our data comes to be missing. That is, we must study the *actual* mechanisms

[21] See `https://scholar.google.com/scholar?cites=16798156444849893273` (as of July 2018). A related paper, Heckman (1979), is claimed to have over 27000 citations (`https://scholar.google.com/scholar?cites=4067958607302478696`), but I am not sure that isn't a record-linkage error in the database.

which lead to missingness. Once we understand them, they will often suggest one, or more, plausible statistical models, which in turn will get us to the $\pi(x, y)$ function we need for our calculations.

Studying this part of the data-generating, or perhaps data-collecting or data-creating, process is a key part of applied statistics, but it's not a *statistical* task in the same way that, say, estimating a regression surface is. Rather, it is something that requires a good deal of substantive, domain-specific knowledge, because the actual causes of missingness are very different in different areas of inquiry. If we are dealing with surveys, for instance, we have to investigate why people[22] don't answer some questions on surveys (but not others). These causes will be very different from those which lead meteorological measuring stations to sometimes fail to record the concentration of certain pollutants in the air. This in turn will have nothing to do with gaps in credit records when making loan decisions.

I.2.5 Causal Inference as a Missing Data Problem, and Vice Versa

The most basic sort of causal inference[23] is asking about the average effect of changing the value of one cause, the **average treatment effect** (ATE):

$$ATE \equiv \mathbb{E}\left[Y|do(X=1)\right] - \mathbb{E}\left[Y|do(X=0)\right] \tag{I.25}$$

If we have many units we can observe, say $i = 1, \ldots n$, we might hope to approximate this by the average of the effects for each unit:

$$ATE \approx \sum_{i=1}^{n} Y_i|do(X=1) - Y_i|do(X=0) \tag{I.26}$$

Unfortunately, we cannot simultaneously give unit i both treatments. Thus one or the other of the two values we want there is un-observed, or, in a word, missing. This is why a lot of work on estimating treatment effects re-uses tools developed for handling missing data, to the point where Donald Rubin, an eminent authority in both areas, has been known to say that "causal inference is a missing data problem" (Ding and Li, 2018).

On the other hand, whether or not some variables are missing has, clearly, some sort of probabilistic connection to those variables. If I have convinced you that you should really investigate in detail *why* variables are missing, you will be led to build models of the missing-data mechanism, and that in turn will lead you to things like graphical models, which make assessing conditional independence relations very straightforward. For instance, the simplest graphical model which would support MAR is given in Figure I.4. From this perspective, whether or not the relationship between Y and X can be identified when Y is sometimes missing turns on what features of the joint distribution are left alone when conditioning

[22] That is, people who can be reached by the survey at all; why some people are easier to survey than others is another, though related, problem.

[23] If you have skipped around, it would be a good idea to read Part III, or at least Chapter 19 before going further in this section.

on M_Y, i.e., on what paths such conditioning opens or closes. One can, therefore, completely reverse the perspective, and treat "missing data as a causal inference problem" (Mohan *et al.*, 2013).

I.3 Further Methods: Imputation, EM, Weighting

Some methods for dealing with missing data can be used whether we assume MAR or informative missingness (though they will give different answers in the two cases). The most important ones are:

- *Making up*, or **imputing**, values for the missing variables, and analyzing the completed data set;
- *Averaging* the log-likelihood function over all possible values of the missing variables, using the EM algorithm;
- *Weighting* the complete observations, so that the point we see appropriate represent the ones we missed.

We will deal with these in turn.

I.3.1 Imputation

An alternative to any form of deletion is to *make something up* for the missing values, and then analyze the data set, with both real and **imputed** values, as though it were complete. The best reason to do such **imputation** is so that the partial cases can be used in procedures (statistical or computational) which require complete data. *Imputation never creates more information*; at best, it uses the available information efficiently, and it can easily lead to systematic distortions. It is thus a tool to be applied carefully. Indeed, I rather suspect half the reason we call this "imputation" is that calling it "making stuff up for the missing values so we can use the data" is so blunt and explicit about what we're doing that it makes people nervous[24].

I.3.1.1 Imputation under MAR

1. *Imputation by hopefully-representative constants* The simplest and oldest sort of imputation replaces every missing value of a variable by the *same* value, derived from the observed cases, usually the mean, median or mode. This is rarely a good idea, because it will distort the relationship between the imputed variable and everything else. In particular, imputing a constant value for a missing response variable will tend to attenuate any regression relationship.
2. *Imputation from the marginal distribution* Rather than using a single constant value, one can also impute *randomly*, using the marginal distribution of observed cases for the missing variable. The oldest form of this replaces each missing value of Y by independently sampling from the observed values of Y.

[24] The other half of the reason is that "making stuff up for the missing values so we can use the data" is a mouthful.

One could also attempt to learn the distribution of Y (by a parametric model, nonparametric density estimation, etc.), and then draw from that.

Sampling from the marginal distribution has the advantage over the typical-value method of not artificially reducing the variance of Y, or otherwise distorting its (marginal) distribution. But, because the draws are independent of other variables, it will tend to attenuate the dependence of Y on those other variables, and so will still introduce distortions into, say, regressions. This would be true even if we were sampling from the true marginal distribution of Y. You should be able to convince yourself that if Y is missing at random, but not missing *completely* at random, then sampling from the observed marginal distribution of Y will introduce distortions. On the other hand, if Y really is missing completely at random, such imputation *only* affects the relationship to other variables.

Figure I.2b is our running example, with marginal-distribution imputation.

3. *Conditional imputation by regression* A somewhat more flexible strategy is to use the complete cases to learn a regression function for the sometimes-missing variable, i.e., to estimate the function $\widehat{\mu} = \mathbb{E}[Y|X]$. In a record where Y is missing and we see $X = x$, then, we impute the value $\widehat{\mu}(x)$ for Y. This tries to preserve something of the relationship between the missing variable and the others, but obviously relies very strongly on missing-ness being uninformative about Y.

   (An additional complication for this strategy is that other variables, besides Y, might be missing for some records but not for others. One thus might have to end up estimating, and using, many different regression functions.)

   Figure I.2c is our running example, with imputation from having smoothed the observed data points.

4. *Conditional imputation from the conditional distribution* Of course, nothing says that we need to only use regression. Once we can estimate $\Pr(Y|X)$, we can draw samples from it, and impute those samples to the missing observations. The easiest form of this is to predict the missing values using regression, and then add noise, but all the techniques used to learn conditional distributions (§14.5) are potentially in play.

   One version of this idea is to impute by *matching*, that is, to search for an observation with the same value of X as the one where Y is missing, and to copy its value of Y; if there are multiple matches, chose among them randomly[25]. The exact performance here will depend on a lot of details — what if there is no match? since exact equality is implausible if X is continuous, how close does it need to be to declare a match? what if X is multi-dimensional?

[25] This is, of course, similar to the matching methods sometimes used in causal inference (§21.1.3).

— which can lead to very complicated algorithms[26]. The point, however, is to sample from $\Pr(Y|X)$.

I.3.1.2 Imputation under Informative Missingness

Basically every idea about imputation under MAR has its counter-part for imputation under informative missingness. The most natural starting point is imputation from the conditional distribution. If we're working under MNAR, and we think we know $\Pr(Y|X, M_Y = 1)$, then we should (in principle) be able to sample from it, and use those sample to fill in the missing values. Figure I.2d shows an example of such imputation, where the trend for the missing values is assumed to run *opposite* to the one we get from the fully-observed data.

In this context, it's worth putting in a word for imputation constants. This can make a fair amount of sense in situations where we know the missing values arose by censoring. If, for instance, we're measuring how long it takes a chemical reaction to run, but each experiment only lasts for a certain amount of time t_0, it can be a good idea to impute $t_0 + \delta$ to all the reactions which didn't finish in time — and to vary δ to see how much that affects our conclusions.

I.3.1.3 Multiple Imputation and Uncertainty

Once we have completed the data set, we can run our usual analyses on it, whatever those might be. When we look at how uncertain those analyses are, however, we really ought to take into account the fact that we did imputation, and we might well have imputed different values than the ones we did. (This is especially true when we impute randomly-sampled values!) One way to get at this is through **multiple imputation**, where we do our stochastic imputation many times, re-running the analysis on each.

At this point, we need to somehow synthesize the multiple results from multiple imputations into a single measure of uncertainty. This is fairly simple if we bootstrap: first draw a bootstrap sample, then do imputation using that sample, and then combine the results as we would ordinarily for bootstrapping.

If we're just reporting means and variances, though, we need to be just a little bit more careful. In these situations, we often calculate a estimate $\hat{\theta}$ from each imputation run, *and* an associated variance σ^2. Say that we get $\hat{\theta}_1, \ldots \hat{\theta}_m$ from our m imputations, and $\sigma_1^2, \ldots \sigma_m^2$ for the variances. Then our best over-all estimate is clearly

$$\overline{\theta} = \frac{1}{m}\sum_{j=1}^{m}\hat{\theta}_j \tag{I.27}$$

[26] A historically important one, developed by the US Census Bureau, was called "hot-deck imputation", because of the way it re-used the punched paper cards then used for data storage. (Using punch cards to store census data actually goes back to the late 1800s, and the tabulating machines were important precursors of digital computers — see, e.g., Yates 1989.) Cards with a similar value of X had just been processed, hence were warm from the card-reading machinery, hence "hot". By contrast, sampling from the marginal distribution by picking a totally random card was "cold-deck imputation". For more on this, see Rubin (1987), and the references therein.

But the associated variance is

$$\frac{1}{m}\sum_{j=1}^{m}\sigma_j^2 + \frac{1}{m-1}\sum_{j=1}^{m}(\hat{\theta}_j - \overline{\theta})^2 \tag{I.28}$$

(This is just our old friend, the law of total variance.)

I.3.2 The EM Algorithm

In §17.2, we looked at the EM algorithm as a way of dealing with *latent* variables, ones which are never observed, but we nonetheless postulate, and imagine are linked to the observable variables in systematic ways. This is very close to the problem we have with missing data, and the EM algorithm can also be used here.

What we would like to do is to maximize the observed-data likelihood

$$p(x, y_{\text{obs}}; \theta) \tag{I.29}$$

By elementary probability,

$$p(x, y_{\text{obs}}; \theta) = \sum_{y} p(x, y_{\text{obs}}, y_{\text{miss}}; \theta) \tag{I.30}$$

i.e., a sum over all possible complete-data likelihoods. (If the missing observations are continuous, make the sum into an integral.) Appealing to the theory in §17.2.1, the EM procedure for doing this is as follows:

1. Start with a guess $\hat{\theta}^{(0)}$ for θ
2. Until things stop improving:
   1. **E-step**: Find the conditional distribution of Y_{miss} given $X = x$ and $Y_{\text{obs}} = y_{\text{obs}}$,
   $$q(y_{\text{miss}}) = p(y_{\text{miss}}|x, y_{\text{obs}}, M_Y; \hat{\theta}^{(t)}) \tag{I.31}$$
   Note that if data points are IID, then $Y_{\text{miss}} \perp\!\!\!\perp Y_{\text{obs}}|X, M_Y$, so this simplifies to just $p(y_{\text{miss}}|x, M_Y; \hat{\theta}^{(t)})$. In fact, with the IID assumption this simplifies even further, since each missing Y value needs to be conditioned only on the corresponding X and its own missing-ness indicator, and the whole joint distribution over missing y values is a product of the distributions for each value.
   2. **M-step**: Find the θ which maximizes the approximation to the complete-data log-likelihood:
   $$\hat{\theta}^{(t+1)} = \operatorname*{argmax}_{\theta} \sum_{y_{\text{miss}}} q(y_{\text{miss}}) \log p(x, y_{\text{obs}}, y_{\text{miss}}, M_Y; theta) \tag{I.32}$$
   Again, if y_{miss} is continuous, replace the sum with an integral.
   If (as in regression problems) we only care about the conditional-on-X likelihood, we use that here, so
   $$\hat{\theta}^{(t+1)} = \operatorname*{argmax}_{\theta} \sum_{y_{\text{miss}}} q(y_{\text{miss}}) \log p(y_{\text{obs}}, y_{\text{miss}}, M_Y|x; \theta) \tag{I.33}$$

Finally, under the IID assumption, both the numerator and the denominator inside the log are products of independent terms for each data point, simplifying the calculation.

3. Return the final $\hat{\theta}^{(t)}$ and (optionally) the final distribution q over the missing observations.

Note that since we are conditioning Y_{miss} not just on X but also M_Y, the EM procedure *can* work when the Y suffers from informative missingness. We just need to guess (correctly) the conditional distribution of missing Y.

It is very likely that this all feels rather abstract. Exercise 9 guides you through implementing the EM algorithm for missing data in a classic problem, both in a version where data are missing at random, and one subject to censoring.

I.3.2.1 Monte Carlo EM

In many situations, the E step of the EM algorithm is hard to implement exactly, because it's difficult to get a closed-form expression for the conditional distribution of the missing observations. In these situations, it is often possible to draw a random *sample* from the distribution instead. Instead of averaging over averaging over $q(y_{\text{miss}})$ in Eq. I.32, then, we average over the sample of different possible y_{miss} values.

I.3.3 Inverse Probability Weighting

A final approach to dealing with missing data is worth mentioning. This is the simple trick of giving more or less weight to the complete observations. Suppose that we a certain data point was fully observed, but we think that its inclusion probability was only 0.1. Then there should have been about nine other data points just like it, in order to get *one* successful observation.

This logic leads to the idea of **inverse probability weighting** (IPW). If we know the inclusion probabilities as a function of X and Y, $\pi(x, y)$, then we give data point i a weight of

$$w_i = \frac{1}{\pi(x_i, y_i)} \tag{I.34}$$

when we compute things like MSE or log-likelihood. This, of course, will simplify if we assume MAR, to just

$$w_i = \frac{1}{\pi(x_i)} \tag{I.35}$$

After that, the analysis proceeds as in Ch. 10. Exercise 10 covers derives from properties for estimating expectation values in this way.

It is worth noting that IPW leans *very* heavily on knowing the inclusion probabilities. This is particularly true when we believe that some of the inclusion probabilities are very small, and so imply very large weights. If we have to es-

timate those probabilities from data[27], then we really ought to propagate the uncertainty from the inclusion probabilities to the sampling weights to our ultimate conclusions. The bootstrap provides a convenient way to do this: in each bootstrap replicate, re-learn the inclusion probabilities, then use those re-learnt weights; pool over replicates as usual.

I.4 Checking Assumptions About Missingness

The sad truth is that most assumptions about missingness cannot be checked in a purely "statistical" way, by running tests on the observed data. This is because most assumptions about missingness are about how Y related to M_Y, and (once more, with feeling) we don't know what Y is when $M_Y = 1$.

The only important exception to this negative conclusion is the assumption of "missing complete at random", MCAR. This is that

$$M_Y \perp\!\!\!\perp X, Y \tag{I.36}$$

This implies that

$$M_Y \perp\!\!\!\perp X \tag{I.37}$$

which only involves observable variables, so we can check it. If we find, in our data, that $M_Y \not\perp\!\!\!\perp X$, we can reject MCAR. Of course, MCAR can be false even if $M_Y \perp\!\!\!\perp X$, because then missingness might still be very informative about Y, and we cannot test this without knowing Y.

Let's turn now to the contrast between missing-at-random, MAR, and informative missingness or missing-not-at-random, MNAR. This is the contrast between $M_Y \perp\!\!\!\perp Y|X$, and $M_Y \not\perp\!\!\!\perp Y|X$. Basic probability tells us that MAR is equivalent to the equation

$$\Pr(Y = y|X = x, M_Y = 1) = \Pr(Y = y|X = x, M_Y = 0) \tag{I.38}$$

holding for all x and y. Unfortunately, our data tells us, quite literally, nothing about $\Pr(Y = y|X = x, M_Y = 1)$. That distribution could be anything at all, and the distribution of what we *observe* would not change[28]. So there is no way to do a formal, statistical test of whether the missing-ness in Y is informative about Y. Whether you take this to mean that the data can never support MAR, or to mean that the data can never undermine MAR, is to some extent a matter of temperament.

However, the fact that there is no formal, statistical test to appeal to does not mean that there is no work for a statistician to do. It is often possible to investigate *why* some data are missing, by a detailed study of the data-collection process. Particular assumptions may also be made more or less plausible by means of analogies with other situations where the missing-data mechanisms are (thought to be) well-understood.

[27] As opposed to, say, knowing them because they reflect properties of the data-collection process under our control.

[28] This doesn't mean that $\Pr(Y = y|X = x)$ can be anything at all, however; see §I.5 below.

I.5 Bounds

So far, all our techniques for handling missing data have focused on somehow working out what the missing values were, or might have been. We might instead frankly accept that we have no idea what they were, but try to place **bounds** on their impact. Think back to Eq. refeqn:complete-data-marginal-prob-in-terms-of-missing-data-marginal-prob:

$$\Pr(Y=y) = \Pr(Y=y|M_Y=0)\,\pi + \Pr(Y=y|M_Y=1)\,(1-\pi) \tag{I.39}$$

We can, in principle, learn $\Pr(Y=y|M_Y=0)$ and π from data — they're identified. $\Pr(Y=y|M_Y=1)$ is *not* identified, without further assumptions, but of course, being a probability, it's between 0 and 1. So we can say, without any assumptions, that

$$\Pr(Y=y|M_Y=0)\,\pi \leq \Pr(Y=y) \leq \Pr(Y=y|M_Y=0)\,\pi + 1 - \pi \tag{I.40}$$

This doesn't tell us everything, but it does rule out some possibilities for $\Pr(Y=y)$ — for instance, it can't be $\frac{1}{2}\Pr(Y=y|M_Y=0)\,\pi$. When the distribution of the data doesn't uniquely determine some quantity, but does put restrictions on it, we say that the quantity is **partially identified** or **set-identified** (as opposed to the usual **point-identified**).

A similar argument gives partial-identification bounds for the conditional probability:

$$\Pr(Y=y|M_Y=0,X=x)\,\pi(x) \leq \Pr(Y=y|X=x) \leq 1+(\Pr(Y=y|M_Y=0,X=x)-1)\pi(x) \tag{I.41}$$

Going further than these bounds typically requires either some detailed consideration of what we're really trying to estimate (when do we care about a single probability?), possibly combined with additional assumptions, e.g., that $\Pr(Y=y|X=x)$ is monotone in y for each x. This sort of elaboration inolves too many special cases to be treated here, but see the references under further reading.

I.6 Closing Modeling Advice

I cannot, unfortunately, provide any hard and fast rules about how to deal with missing data. I can, however, provide some advice.

1. The best way to deal with missing data is not to have any. To the extent that you can influence how the data are collected, try to make sure that missing data does not arise; failing that, try to make sure that missingness is uninformative and rare. It's usually better to spend your efforts securing good data to begin with, than trying to compensate for bad data collection later with fancy techniques[29].

[29] Statisticians get rewarded, professionally, for developing new techniques, so that's the focus of most of our scientific literature on missing data. But I don't think any experienced applied statistician would disagree with this point.

2. Understand, concretely, why some of *this* data might go missing. If you can change how the data is collected, this will help you make sure less does go missing (as in the first point). If the data are just handed to you and you have to make the best of them, then this understand is crucial to creating stochastic models of missingness, and deciding whether MAR is plausible or not.
3. Consider more than one model for missingness. Even if you think MAR is plausible (or at least defensible), it's usually a good idea to consider at least one or two models of informative missingness. If small departures from your initial or most-favorite model don't change the conclusions of your analysis very much, well and good. (Though, in that case, you might want to ask *how big* a departure would be needed to seriously alter a conclusion.) If, on the other hand, your conclusions are very sensitive to exactly how you deal with missing values, you either need a strong justification for preferring one model over another, or you need to be up-front about just how much your conclusions rest on assumptions about missing data.

I.7 Further Reading

The classic reference on missing data, which standardized the MCAR/MAR/MNAR terminology, is Little and Rubin (1987). From the same school, Rubin (1987) is a classic reference on multiple imputation.

On the EM algorithm, in addition to the references given in Ch. 17, McLachlan and Krishnan (2008) includes an extensive treatment of its uses in missing data problems. The Monte Carlo EM approach, where in the E step we *sample* from the distribution of missing values instead of *calculating* that distribution, still requires us to iterate the E and M steps many times. Sung and Geyer (2007) provides a truly ingenious procedure where we need only a single, data-independent sample of potential missing values. The basic idea of the paper is well within the grasp of readers of this book, though the proofs of its validity are much more technical.

The bounding approach, as an alternative to identifying assumptions, is most closely associated with the work of Charles Manski. Manski (2007) is the easiest introduction to his thought; see especially Chapter 2 of that book. Manski (2003) is a more thoroughly technical treatment.

Becker (2017) provides a rich discussion of many of the ways data-gathering on social phenomena can go wrong, including highly-informative missingness, and case studies of how to investigate such problems. Much of his discussion applies equally well to measurement in other domains.

I.8 Exercises

1. In this exercise, assume that Y is missing at random, but not missing completely at random, and that X is not missing at all.
    1. Show that the complete-data joint distribution, $\Pr(X = x, Y = y)$, is not equal to the observed joint distribution, $\Pr(X = x, Y = y | M_Y = 0)$.

   2. Show that the complete-data joint distribution is still observationally identified.
   3. Can the complete-data joint distribution be estimated *after* listwise deletion?

2. In this exercise, assume that Y is missing at random, but not missing completely at random, and that X is not missing at all.
   1. Show that $\Pr(Y = y|M_Y = 0) \neq \Pr(Y = y)$, so that the observed marginal distribution of Y is not equal to the complete-data marginal distribution.
   2. Is $\Pr(Y = y)$ identified?

3. Read §18.4 on mutual and conditional information.
   1. Show that MCAR holds if and only if $I[Y; M_Y] = 0$
   2. Show that MAR holds if and only if $I[Y; M_Y|X] = 0$.
   3. Show that MNAR holds if and only if $I[Y; M_Y|X] > 0$.
   4. Explain why MNAR is also called "informative missingness".

4. Refer to Figure I.5 (and to the concept of d-separation in Chapter 18).
   1. Use the d-separation rules to explain why

$$Y \perp\!\!\!\perp M_Y|\{V, W, Z, R\} \tag{I.42}$$

      but

$$Y \not\perp\!\!\!\perp M_Y|\{V, W, Q\} \tag{I.43}$$

   2. Decision-makers often rely on proxies for the variables they really wish they had. In the diagram, R could be such a proxy, since it's not a causal ancestor of Y. Does this mean that we could still have $Y \perp\!\!\!\perp M_Y|\{V, W, Z\}$? Explain.
   3. Can you find a set of variables S such that S does not include all the parents of M_Y, but $Y \perp\!\!\!\perp M_Y|S$? If so, give it, and explain why the independence holds; if not, explain why such a set is impossible.

5. Derive Eq. I.18.
6. Derive Eq. I.19.
7. Show that Y is MAR if and only if $\pi(x, y) = \pi(x)$. Explain how Eq. I.19 can still apply when Y is missing-at-random, and why MAR is also known as "ignorable missingness".
8. In both parts of this exercise, but *only* in this exercise, make all the assumptions of §I.2.3.1. *Note:* The first part is (for most people!) much easier than the second.
   1. Find the variance of $\epsilon - \eta$ in terms of the elements of the variance matrix $\mathbf{\Sigma}$.
   2. Derive Eq. I.23.
   3. Find the covariance of ϵ and $\frac{\eta - \epsilon}{\sigma}$ in terms of the elements of $\mathbf{\Sigma}$.
   4. Derive Eq. I.24.

9. *Detailed example of EM for missing data* In this problem, we work through the EM algorithm for missing data, in the classic (i.e., simple!) case of exponentially-distributed random variables. Each unit i has a lifetime Y_i, and these follow an exponential distribution, so the PDF is $\theta e^{-\theta y}$. Assume throughout that the Y_i are independent and identically distributed.
    1. Assume all the Y_i are observed. Write out the log-likelihood and find the MLE of θ.
    2. Assume that some of the Y_i are missing completely at random, with probability ρ.
        1. Write out the log-likelihood for the observed values of Y and the missingness indicators.
        2. Find the MLE for θ and ρ, based on this log-likelihood for observables alone.
        3. (E-step) Find the conditional distribution for the unobserved values of Y, given the observed values and the missingness indicators. This should be a function of θ. (Should it also be a function of ρ?)
        4. (M-step) Write out the complete-data log-likelihood.
        5. (M-step, continued) Write out the expected value of the complete data log-likelihood, averaging over the distribution of missing values. *Hint:* What is the expected value of an exponential distribution?
        6. Find an expression for $\hat{\theta}^{(t+1)}$ in terms of $\hat{\theta}^{(t)}$ and the data.
        7. Find the fixed point of this expression. Does it match the MLE for θ you found earlier?
    3. Assume that the Y_i are *censored*: there is a time t_0 such that if $Y_i \leq t_0$, we get to see Y_i, but if $Y_i > t_0$, Y_i is missing (and we know that it's missing). Assume we know t_0.
        1. Write out the log-likelihood function based on the observed Y_i and the missingness indicators. *Hint:* The probability of an observation being censored is a function of θ and t_0.
        2. Find the maximum likelihood estimator of θ based on this observed-data log-likelihood.
        3. (E-step) Find the conditional distribution of the missing Y_i, given the observed Y_i and the missingness indicators. *Hint:* If $Y \sim \text{Exp}(\theta)$, then $Y|Y > y_0$ follows what distribution?
        4. (M-step) Write out the complete-data log-likelihood.
        5. (M-step) Write out the expected value of the complete-data log-likelihood, averaging over the distribution of missing values.
        6. Find an expression for $\hat{\theta}^{(t+1)}$ in terms of $\hat{\theta}^{(t)}$ and the data.
        7. Find the fixed point of this expression. Does it match the MLE for θ you found earlier?
        8. Explain what assumption this procedure makes about the unobserved values of Y, and why this assumption cannot be tested (with this data).
        9. Can you extend the procedure to handle the case where each unit i has its own (known) censoring time t_i?

10. A classic example of inverse probability weighting concerns estimating the mean (or total) of a population from samples. Suppose that there are n members of the population, each with some value of Y, say Y_i. The population mean is therefore

$$\overline{y} = \frac{1}{n}\sum_{i=1}^{n} Y_i \tag{I.44}$$

We actually observe $N < n$ members of the population, say the ones where $i \in O$. The probability of observing Y_i is π_i. The **Horvitz-Thompson** estimate of the population mean is

$$\hat{y}_{HT} = \frac{1}{n}\sum_{i\in O} \frac{Y_i}{\pi_i} \tag{I.45}$$

Note that the denominator is the *total* population size, *not* the sample size!

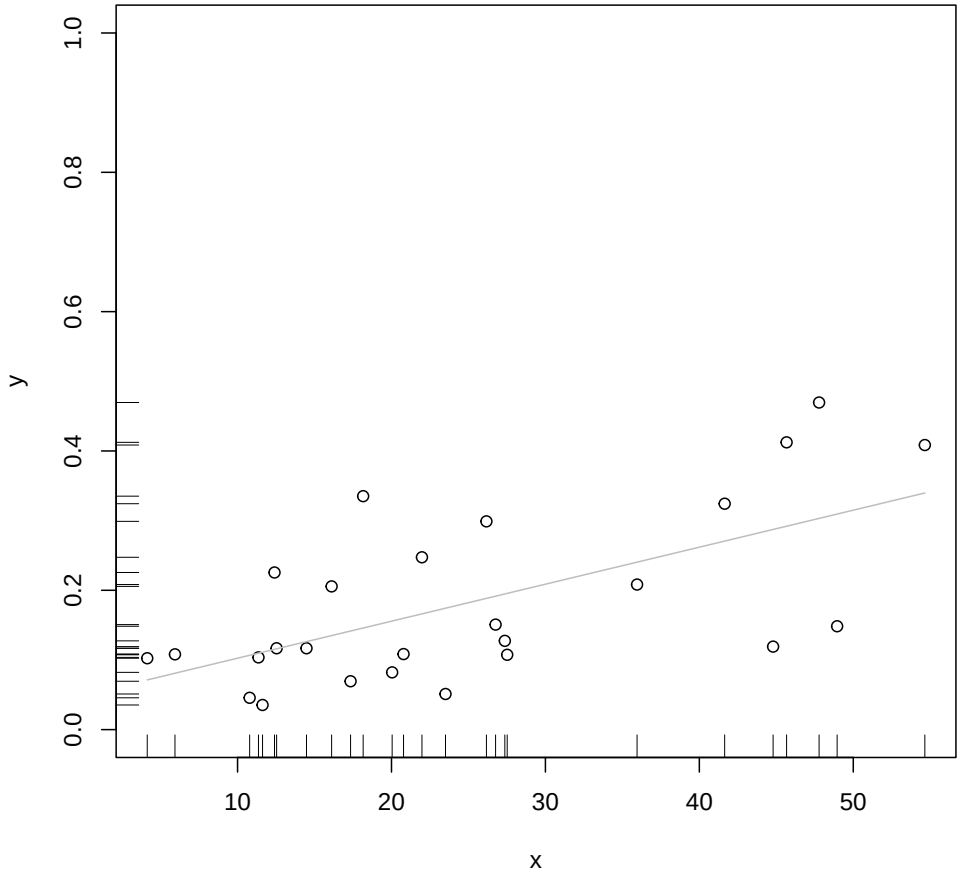
1. Show that $\hat{y}_{HT}$ is an unbiased estimate of $\overline{y}$. *Hint:* First show that $\sum_{i\in O} Y_i = \sum_{i=1}^{n} Y_i(1 - M_{Y_i})$.
2. Find the variance of $\hat{y}_{HT}$. You will need the joint probability that both i and j are observed, π_{ij}.
3. Show that the variance $\to 0$ as N grows.

The Running Example

Here is the code for the simulation that provides this chapter's running example:

```
library(faraway) # for ilogit
n <- 50
# X is uniform (put it in order for easy plotting)
x <- sort(runif(n, min=0, max=100))
# Y increases with X, though non-linearly
y <- ilogit(0.05*(x-50)+rnorm(n, sd=1))
# Missing-ness depends on the value of Y, high values => more missing
prob.y.missing <- ilogit(50*logit(y))
missing.y <- (rbinom(n=n, size=1, prob=prob.y.missing) == 1) # To make it Boolean
y.obs <- y[!missing.y]
x.obs <- x[!missing.y]
the.df <- data.frame(x=x, y=ifelse(missing.y, NA, y), missing.y=missing.y)
plot(y~x, data=the.df, xlab="x", ylab="y", ylim=c(0,1))
rug(side=1, x=the.df$x)
rug(side=2, x=the.df$y)
```

As you may have worked out by a process of elimination, this is shown in Figure I.2a. Missingness is here directly based on Y, and so informative. Notice, by the way, that when we run a regression on the fully-observed data points (as in Figure I.2c, or Figure I.3), we get a very different regression curve than is implied by the actual generative process.




```
the.df.post.deletion <- na.omit(the.df)
plot(y ~ x, data=the.df.post.deletion, xlab="x", ylab="y", type="p",
     xlim=c(min(x), max(x)), ylim=c(0,1))
rug(side=1, x=the.df.post.deletion$x)
rug(side=2, x=the.df.post.deletion$y)
require(mgcv)
a.spline <- gam(y ~ s(x), data=the.df.post.deletion)
lines(the.df.post.deletion$x, fitted(a.spline), col="grey")
```

Figure I.3 What the running-example data looks like, after deleting incomplete cases. The grey line is a spline run through the fully-observed points.

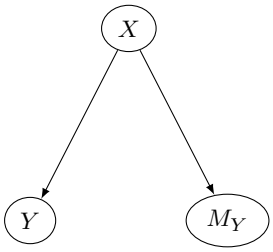


Figure I.4 The simplest (but not the only!) graphical model in which Y might be missing at random (MAR), but not missing completely at random (MCAR).

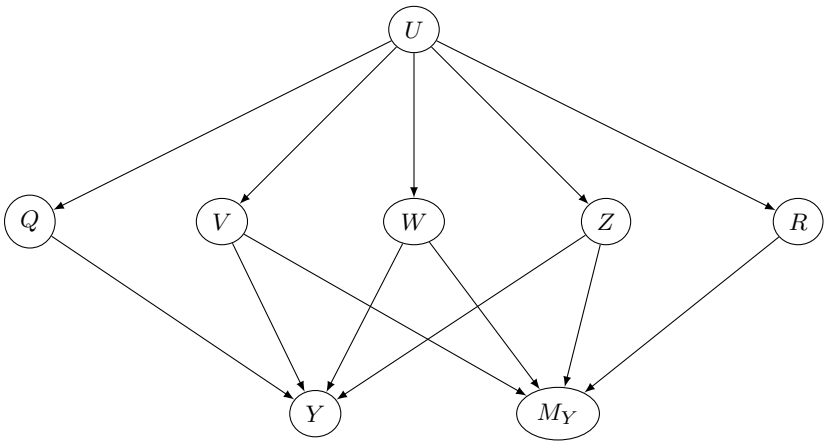


Figure I.5 Whether Y is missing-at-random or not can depend on the variables used for conditioning. Suppose that whether or not a student is admitted (or a loan approved, or an arrestee released) depends on $X = \{V, W, Z, R\}$, and Y is some measure of academic success (or loan repayment, or subsequent trouble with the law). Then $Y \perp\!\!\!\perp M_Y | X$, and Y is missing-at-random for the decision-makers. But we as data-analysts might only have access to $X' = \{V, W, Q\}$, and then $Y \not\perp\!\!\!\perp M_Y | X'$. See Exercise 4 for proofs and extensions.

Appendix J

Writing R Functions

The ability to read, understand, modify and write simple pieces of code is an essential skill for modern data analysis. Lots of high-quality software already exists for specific purposes, which you can and should use, but statisticians need to grasp how such software works, tweak it to suit their needs, recombine existing pieces of code, and, when needed, build their own tools. Someone who just knows how to run canned routines is not a data analyst but a human interface to a machine they do not understand.

Fortunately, writing code is not actually very hard, especially not in R. All it demands is the discipline to think logically, and the patience to practice. This appendix tries to illustrate what's involved, starting from the very beginning. It is redundant for many students, but included through popular demand.

J.1 Functions

Programming in R is organized around **functions**. You all know what a mathematical function is, like $\log x$ or $\phi(z)$ or $\sin\theta$: it is a rule which takes some **inputs** and delivers a definite **output**. A function in R, like a mathematical function, takes zero or more inputs, also called **arguments**, and **returns** an output. The output is arrived at by going through a series of calculations, based on the input, which we specify in the body of the function. As the computer follows our instructions, it may do other things to the system; these are called **side-effects**. (The most common sort of side-effect, in R, is probably making or updating a plot on the screen.) The basic **declaration** or **definition** of a function looks like so:

```
my.function <- function(argument.1, argument.2, ...) {
  # clever manipulations of arguments
  return(the.return.value)
}
```

Strictly speaking, we often don't need the `return()` command; without it, the function will return the last thing it evaluated. But it's usually clearer, and never hurts, to be explicit.

We write functions because we often find ourselves going through the same sequence of steps at the command line, perhaps with small variations. It saves mental effort on our part to take that sequence and bind it together into an

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

integrated procedure, the function, so that then we can think about the function as a whole, rather than the individual steps. It also reduces error, because, by invoking the same function every time, we don't have to worry about missing a step, or wondering whether we forgot to change the third step to be consistent with the second, and so on.

J.2 First Example: Pareto Quantiles

Let me give a really concrete example. In Chapter 6, I mentioned the **Pareto distribution**, which has the probability density function

$$f(x;\alpha,x_0) = \begin{cases} \frac{\alpha-1}{x_0}\left(\frac{x}{x_0}\right)^{-\alpha} & x \geq x_0 \\ 0 & x < x_0 \end{cases} \tag{J.1}$$

Consequently, the CDF is

$$F(x;\alpha,x_0) = 1 - \left(\frac{x}{x_0}\right)^{-\alpha+1} \tag{J.2}$$

and the quantile function is

$$Q(p;\alpha,x_0) = x_0(1-p)^{-\frac{1}{\alpha-1}} \tag{J.3}$$

Say I want to find the median of a Pareto distribution with $\alpha = 2.33$ and $x_0 = 6 \times 10^8$. I can do that in R:

```
6e8 * (1-0.5)^(-1/(2.33-1))
## [1] 1010391288
```

If I decide I want the 40th percentile of the same distribution, I can do that:

```
6e8 * (1-0.4)^(-1/(2.33-1))
## [1] 880957225
```

If I decide to raise the exponent to 2.5, lower the threshold to 1×10^6, and ask about the 92nd percentile, I can do that, too:

```
1e6 * (1-0.92)^(-1/(2.5-1))
## [1] 5386087
```

But doing this all by hand gets quite tiresome, and at some point I'm going to mess up and (say) type when I meant ^. I'll write a function to do this for me, and so that there is only *one* place for me to make a mistake:

```
# Calculate quantiles of the Pareto distribution
# Inputs: desired quantile (p)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
# Outputs: the pth quantile
qpareto.1 <- function(p, exponent, threshold) {
  q <- threshold*((1-p)^(-1/(exponent-1)))
  return(q)
}
```

The name of the function is what goes on the left of the assignment `<-`, with the declaration (beginning `function`) on the right. (I called this `qpareto.1` to distinguish it from later modifications.) The three terms in the parenthesis after `function` are the arguments to `qpareto` — the inputs it has to work with. The body of the function is just like some R code we would type into the command line, after assigning values to the arguments. The very last line tells the function, explicitly, what its output or return value should be. Here, of course, the body of the function calculates the pth quantile of the Pareto distribution with the exponent and threshold we ask for.

When I enter the code above, defining `qpareto.1`, into the command line, R just accepts it without outputting anything. It thinks of this as assigning certain value to the name `qpareto.1`, and it doesn't produce outputs for assignments when they succeed, just as if I'd said `alpha <- 2.5`.

All that successfully creating a function means, however, is that we didn't make a huge error in the syntax. We should still check that it works, by invoking the function with values of the arguments where we know, by other means, what the output should be. I just calculated three quantiles of Pareto distributions above, so let's see if we can reproduce them.

```
qpareto.1(p=0.5,exponent=2.33,threshold=6e8)
## [1] 1010391288
qpareto.1(p=0.4,exponent=2.33,threshold=6e8)
## [1] 880957225
qpareto.1(p=0.92,exponent=2.5,threshold=1e6)
## [1] 5386087
```

So, our first function seems to work successfully.

J.3 Functions Which Call Functions

If we examine other quantile functions (e.g., `qnorm`), we see that most of them take an argument called `lower.tail`, which controls whether p is a probability from the lower tail or the upper tail. `qpareto.1` implicitly assumes that it's the lower tail, but let's add the ability to change this.

```
# Calculate quantiles of the Pareto distribution
# Inputs: desired quantile (p)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
  # flag for whether to give lower or upper quantiles (lower.tail)
# Outputs: the pth quantile
qpareto.2 <- function(p, exponent, threshold, lower.tail=TRUE) {
  if(lower.tail==FALSE) {
    p <- 1-p
  }
  q <- threshold*((1-p)^(-1/(exponent-1)))
  return(q)
}
```

When, in a function declaration, an argument is followed by `=` and an expression, the expression sets the **default value** of the argument, the one which will be used unless explicitly over-ridden. The default value of `lower.tail` is TRUE, so, unless it is explicitly set to false, we will assume `p` is a probability counted from $-\infty$ on up.

The `if` command is a **control structure** — if the condition in parenthesis is true, then the commands in the following braces will be executed; if not, not. Since lower tail probabilities plus upper tail probabilities must add to one, if we are given an upper tail probability, we just find the lower tail probability and proceed as before.

Let's try it:

```
qpareto.2(p=0.5,exponent=2.33,threshold=6e8,lower.tail=TRUE)
## [1] 1010391288
qpareto.2(p=0.5,exponent=2.33,threshold=6e8)
## [1] 1010391288
qpareto.2(p=0.92,exponent=2.5,threshold=1e6)
## [1] 5386087
qpareto.2(p=0.5,exponent=2.33,threshold=6e8,lower.tail=FALSE)
## [1] 1010391288
qpareto.2(p=0.92,exponent=2.5,threshold=1e6,lower.tail=FALSE)
## [1] 1057162
```

First: the answer `qpareto.2` gives with `lower.tail` explicitly set to true matches what we already got from `qpareto.1`. Second and third: the default value for `lower.tail` works, and it works for two different values of the other arguments. Fourth and fifth: setting `lower.tail` to `FALSE` works properly (since the 50th percentile is the same from above or from below, but the 92nd percentile is different, and smaller from above than from below).

The function `qpareto.2` is equivalent to this:

```
# Calculate quantiles of the Pareto distribution
# Inputs: desired quantile (p)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
  # flag for whether to give lower or upper quantiles (lower.tail)
# Outputs: the pth quantile
qpareto.3 <- function(p, exponent, threshold, lower.tail=TRUE) {
  if(lower.tail==FALSE) {
    p <- 1-p
  }
  q <- qpareto.1(p, exponent, threshold)
  return(q)
}
```

When R tries to execute this, it will look for a function named `qpareto.1` in the workspace. If we have already defined such a function, then R will execute it, with the arguments we have provided, and `q` will become whatever is returned by `qpareto.1`. When we give R the above function definition for `qpareto.3`, it does not check whether `qpareto.1` exists — it only has to be there at run time. If

`qpareto.1` changes, then the behavior of `qpareto.3` will change with it, *without our having to redefine* `qpareto.3`.

This is *extremely useful*. It means that we can take our programming problem and sub-divide it into smaller tasks efficiently. If I made a mistake in writing `qpareto.1`, when I fix it, `qpareto.3` automatically gets fixed as well — along with any other function which calls `qpareto.1`, or `qpareto.3` for that matter. If I discover a more efficient way to calculate the quantiles and modify `qpareto.1`, the improvements are likewise passed along to everything else. But when I *write* `qpareto.3`, I don't have to worry about how `qpareto.1` works, I can just assume it does what I need somehow.

J.3.1 Sanity-Checking Arguments

It is good practice, though not *strictly* necessary, to write functions which check that their arguments make sense before going through possibly long and complicated calculations. For the Pareto quantile function, for instance, p must be in $[0, 1]$, the exponent α must be at least 1, and the threshold x_0 must be positive, or else the mathematical function just doesn't make sense.

Here is how to check all these requirements:

```
# Calculate quantiles of the Pareto distribution
# Inputs: desired quantile (p)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
  # flag for whether to give lower or upper quantiles (lower.tail)
# Outputs: the pth quantile
qpareto.4 <- function(p, exponent, threshold, lower.tail=TRUE) {
  stopifnot(p >= 0, p <= 1, exponent > 1, threshold > 0)
  q <- qpareto.3(p,exponent,threshold,lower.tail)
  return(q)
}
```

The function `stopifnot` halts the execution of the code, with an error message, if all of its arguments do not evaluate to `TRUE`. If all those conditions are met, however, R just goes on to the next command, which here happens to be running `qpareto.3`. Of course, I could have written the checks on the arguments directly into the latter.

Let's see this in action:

```
qpareto.4(p=0.5,exponent=2.33,threshold=6e8,lower.tail=TRUE)
## [1] 1010391288
qpareto.4(p=0.92,exponent=2.5,threshold=1e6,lower.tail=FALSE)
## [1] 1057162
qpareto.4(p=1.92,exponent=2.5,threshold=1e6,lower.tail=FALSE)
```

```
## Error in qpareto.4(p = 1.92, exponent = 2.5, threshold = 1e+06, lower.tail = FALSE):
p <= 1 is not TRUE
```

```
qpareto.4(p=-0.02,exponent=2.5,threshold=1e6,lower.tail=FALSE)
```

```
## Error in qpareto.4(p = -0.02, exponent = 2.5, threshold = 1e+06, lower.tail = FALSE):
p >= 0 is not TRUE
```

```
qpareto.4(p=0.92,exponent=0.5,threshold=1e6,lower.tail=FALSE)
```

```
## Error in qpareto.4(p = 0.92, exponent = 0.5, threshold = 1e+06, lower.tail = FALSE):
exponent > 1 is not TRUE
```

```
qpareto.4(p=0.92,exponent=2.5,threshold=-1,lower.tail=FALSE)
```

```
## Error in qpareto.4(p = 0.92, exponent = 2.5, threshold = -1, lower.tail = FALSE):
threshold > 0 is not TRUE
```

```
qpareto.4(p=-0.92,exponent=2.5,threshold=-1,lower.tail=FALSE)
```

```
## Error in qpareto.4(p = -0.92, exponent = 2.5, threshold = -1, lower.tail = FALSE):
p >= 0 is not TRUE
```

The first two lines give the same results as our earlier functions — as they should, because all the arguments are in the valid range. The third, fourth, fifth and sixth lines all show that `qpareto.4` stops with an error message when one of the conditions in the `stopifnot` is violated. Notice that the error message says *which* condition was violated. The seventh line shows one limitation of this: the arguments violate *two* conditions, but `stopifnot`'s error message will only mention the *first* one. (What is the other violation?)

J.4 Layering Functions and Debugging

Functions can call functions which call functions, and so on indefinitely. To illustrate, I'll write a function which generates Pareto-distributed random numbers, using the "quantile transform" method from §5.2.2.3. This first generates a uniform random number U on $[0, 1]$, and then returns $Q(U)$, with Q being the quantile function of the desired distribution.

The first version contains a deliberate bug, which I will show how to track down and fix.

```
# Generate random numbers from the Pareto distribution
# Inputs: number of random draws (n)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
# Outputs: vector of random numbers
rpareto <- function(n,exponent,threshold) {
  x <- vector(length=n)
  for (i in 1:n) {
    x[i] <- qpareto.4(p=rnorm(1),exponent=exponent,threshold=threshold)
  }
  return(x)
}
```

Notice that this calls `qpareto.4`, which calls `qpareto.3`, which calls `qpareto.1`. It doesn't work:

```
rpareto(10)
```

```
## Error in qpareto.4(p = rnorm(1), exponent = exponent, threshold = threshold): p
>= 0 is not TRUE
```

This is a puzzling error message — the expression `exponent` > 1 never appears in `rpareto`! The error is coming from further down the chain of execution. We can see where it happens by using the `traceback()` function, which gives the chain of function calls leading to the latest error[1]:

```
<<wrapper=TRUE, eval=FALSE>>=

traceback()
## 3: stopifnot(p >= 0, p <= 1, exponent > 1, threshold > 0) at #2
## 2: qpareto.4(p = rnorm(1), exponent = exponent, threshold = threshold) at #4
## 1: rpareto(10)

@
```

`traceback()` outputs the sequence of function calls leading up to the error in reverse order, so that the last line, numbered `1`, is what we actually entered on the command line. This tells us that the error is happening when `qpareto.4` tries to check the arguments to the quantile function. And the reason it is happening is that we are not providing `qpareto.4` with any value of `exponent`. And the reason *that* is happening is that we didn't give `rpareto` any value of `exponent` as an explicit argument when we called it, and our definition didn't set a default.

Let's try this again.

```
rpareto(n=10,exponent=2.5,threshold=1)
```

```
## Error in qpareto.4(p = rnorm(1), exponent = exponent, threshold = threshold): p
>= 0 is not TRUE
```

```
<<wrapper=TRUE, eval=FALSE>>=

traceback()
## 3: stopifnot(p >= 0, p <= 1, exponent > 1, threshold > 0) at #2
## 2: qpareto.4(p = rnorm(1), exponent = exponent, threshold = threshold) at #4
## 1: rpareto(n = 10, exponent = 2.5, threshold = 1)

@
```

This is progress! The `stopifnot` in `qpareto.4` is at least able to evaluate all the conditions — it just happens that one of them is false. The problem, then, is that `qpareto.4` is being passed a negative value of `p`. This tells us that the problem is coming from the part of `rpareto.1` which sets `p`. Looking at that,

```
p = rnorm(1)
```

the culprit is obvious: I stupidly wrote `rnorm`, which generates a *Gaussian* random number, when I meant to write `runif`, which generates a *uniform* random number.[2]

The obvious fix is just to replace `rnorm` with `runif`:

[1] For users of knitr/R Markdown: `traceback` is one of a number of highly-interactive commands which don't work properly in knitr. This is not much of a loss, since it's for debugging, and you shouldn't be doing your debugging in your report.

[2] I actually made this exact mistake the first time I wrote the function, in 2004.

```
# Generate random numbers from the Pareto distribution
# Inputs: number of random draws (n)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
# Outputs: vector of random numbers
rpareto <- function(n,exponent,threshold) {
  x <- vector(length=n)
  for (i in 1:n) {
    x[i] <- qpareto.4(p=runif(1),exponent=exponent,threshold=threshold)
  }
  return(x)
}
```

Let's see if this is enough to fix things, or if I have any other errors:

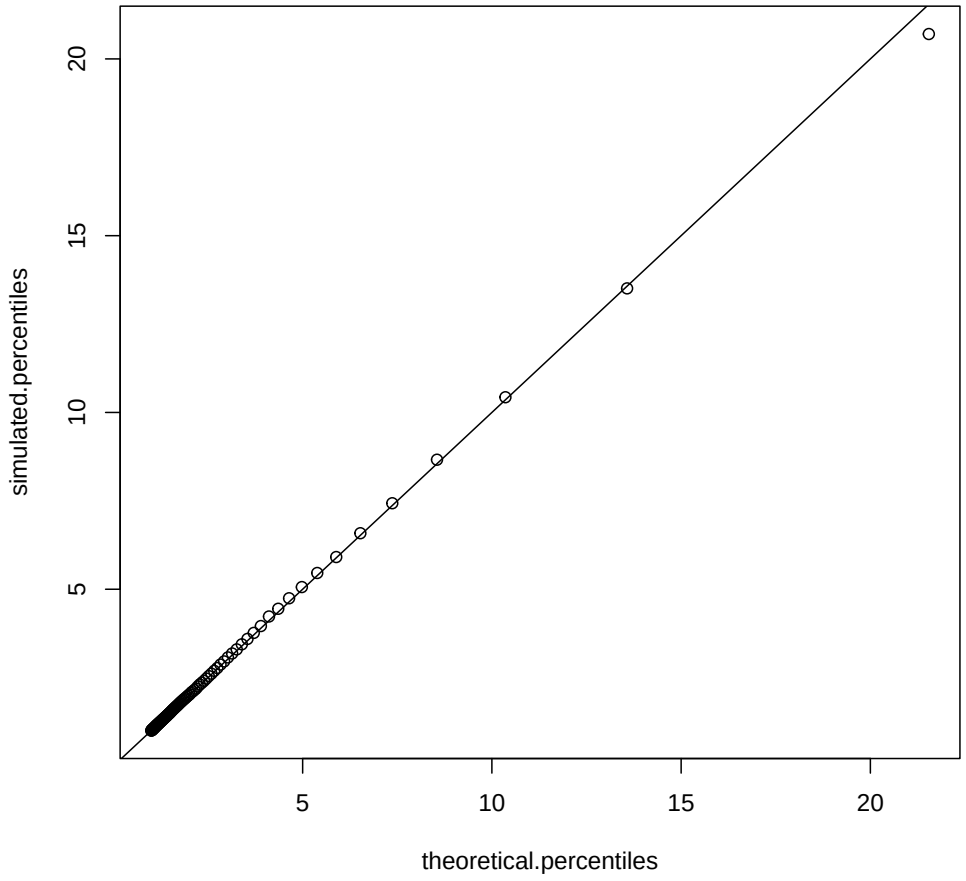
```
rpareto(n=10,exponent=2.5,threshold=1)
##  [1] 1.065780 7.478095 1.098797 1.646324 2.183139 4.160393 1.778903 4.145061
##  [9] 2.166091 1.300081
```

This function at least produces numerical return values rather than errors! Are they the right values?

We can't expect a random number generator to always give the same results, so I can't cross-check this function against direct calculation, the way I could check `qpareto.1`. (Actually, one way to check a random number generator is to make sure it *doesn't* give identical results when run twice!) It's at least encouraging that all the numbers are above `threshold`, but that's not much of a test. However, since this *is* a random number generator, if I use it to produce a lot of random numbers, the quantiles of the output should be close to the theoretical quantiles, which I *do* know how to calculate.

```
r <- rpareto(n=1e4,exponent=2.5,threshold=1)
qpareto.4(p=0.5,exponent=2.5,threshold=1)
## [1] 1.587401
quantile(r,0.5)
##      50%
## 1.609456
qpareto.4(p=0.1,exponent=2.5,threshold=1)
## [1] 1.072766
quantile(r,0.1)
##      10%
## 1.074571
qpareto.4(p=0.9,exponent=2.5,threshold=1)
## [1] 4.641589
quantile(r,0.9)
##      90%
## 4.744159
```

This looks pretty good. Figure J.1 shows a plot comparing all the theoretical percentiles to the simulated ones, confirming that we didn't just get lucky with choosing particular percentiles above.




```
simulated.percentiles <- quantile(r,(0:99)/100)
theoretical.percentiles <- qpareto.4((0:99)/100,exponent=2.5,threshold=1)
plot(theoretical.percentiles,simulated.percentiles)
abline(0,1)
```

Figure J.1 Theoretical percentiles of the Pareto distribution with $\alpha = 2.5$, $x_0 = 1$, and empirical percentiles from a sample of 10^4 values simulated from it with the `rpareto` function. (The solid line is the $x = y$ diagonal, for visual reference.)

J.4.1 More on Debugging

Everyone who writes their own code spends a lot of time debugging[3]. There are some guidelines for making it easier and less painful.

Characterize the Bug

We've got a bug when the code we've written won't do what we want. To fix this, it helps a lot to know exactly what error we're seeing. The first step to this is to make the error reproducible. Can we always get the error when re-running the same code and values? If we start the same code in a clean copy of R, does the same thing happen? Once we can reproduce the error, we map its boundaries. How much can we change the inputs and get the same error? A different error? For what inputs (if any) does the bug go away? How big is the error?

Localize the Bug

The problem *may* be a diffuse all-pervading wrongness, but often it's a lot more localized, to a few lines or even just one line of code; it helps to know where! We have seen some tools for localizing the bug above: `traceback()` and `stopifnot()`. Another very helpful one is to add `print` statements, so that our function gives us messages about the progress of its calculations, selected variables, etc., as it goes; the `warning` command can be used to much the same effect[4].

Fix the Bug

Once you know what's going wrong and where it's going wrong, it's often not too hard to spot the error, either one of syntax (say `=` vs. `==`) or logic. Try a fix and see if it makes it better. Do the inputs which gave you the bugs before now work properly? Are you getting different errors?

J.5 Automating Repetition and Passing Arguments

The match between the theoretical quantiles and the simulated ones in Figure J.1 is close, but it's not perfect. On the one hand, this might indicate some subtle mistake. On the other hand, it might just be random sampling noise — `rpareto` *is* supposed to be a random number generator, after all. We could check this by seeing whether we get *different* deviations around the line with different runs of `rpareto`, or if on the contrary they all pull in the same direction. We could just make many plots by hand, the way we made that plot by hand, but since we're doing almost exactly the same thing many times, let's write a function.

```
# Compare random draws from Pareto distribution to theoretical quantiles
# Inputs: None
# Outputs: None
```

[3] Those who don't write their own code but use computers anyway spend a lot of time putting up with other people's bugs.

[4] Real software engineers look down on this, in favor of more sophisticated tools, like interactive debuggers. They have a point, but that's usually over-kill for the purposes of this class.

```
# Side-effects: Adds points showing random draws vs. theoretical quantiles
  # to current plot
pareto.sim.vs.theory <- function() {
  r <- rpareto(n=1e4,exponent=2.5,threshold=1)
  simulated.percentiles <- quantile(r,(0:99)/100)
  points(theoretical.percentiles,simulated.percentiles)
}
```

This doesn't return anything. All it does is draw a new sample from the same Pareto distribution as before, re-calculate the simulated percentiles, and add them to an existing plot — this is an example of a side-effect. Notice also that the function presumes that `theoretical.percentiles` already exists. (The theoretical percentiles won't need to change from one simulation draw to the next, so it makes sense to only calculate them once.)

Figure J.2 shows how we can use it to produce multiple simulation runs. We can see that, looking over many simulation runs, the quantiles seem to be too large about as often, and as much, as they are too low, which is reassuring.

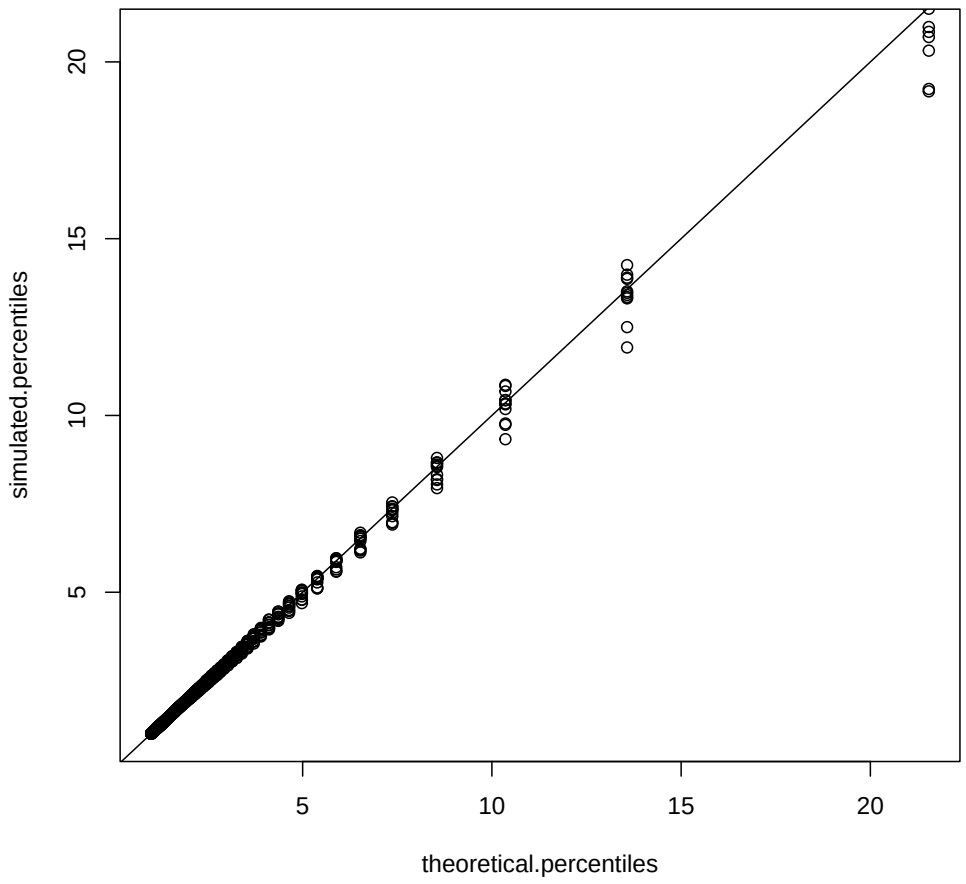
One thing which that figure doesn't do is let us trace the connections between points from the same simulation. More generally, we can't modify the plotting properties, which is kind of annoying. This is easily fixed modifying the function to **pass along arguments**:

```
# Compare random draws from Pareto distribution to theoretical quantiles
# Inputs: Graphical arguments, passed to points()
# Outputs: None
# Side-effects: Adds points showing random draws vs. theoretical quantiles
  # to current plot
pareto.sim.vs.theory <- function(...) {
  r <- rpareto(n=1e4,exponent=2.5,threshold=1)
  simulated.percentiles <- quantile(r,(0:99)/100)
  points(theoretical.percentiles,simulated.percentiles,...)
}
```

Putting the ellipses (`...`) in the argument list means that we can give `pareto.sim.vs.theory.2` an arbitrary collection of arguments, but with the expectation that it will pass them along unchanged to some other function that it will call with `...` — here, that's the `points` function. Figure J.3 shows how we can use this, by passing along graphical arguments to `points` — in particular, telling it to connect the points by lines (`type="b"`), varying the shape of the points (`pch=i`) and the line style (`lty=i`).

These figures are reasonably convincing that nothing is going seriously wrong with the simulation for *these* parameter values. To check other parameter settings, again, I could repeat all these steps by hand, or I could write another function:

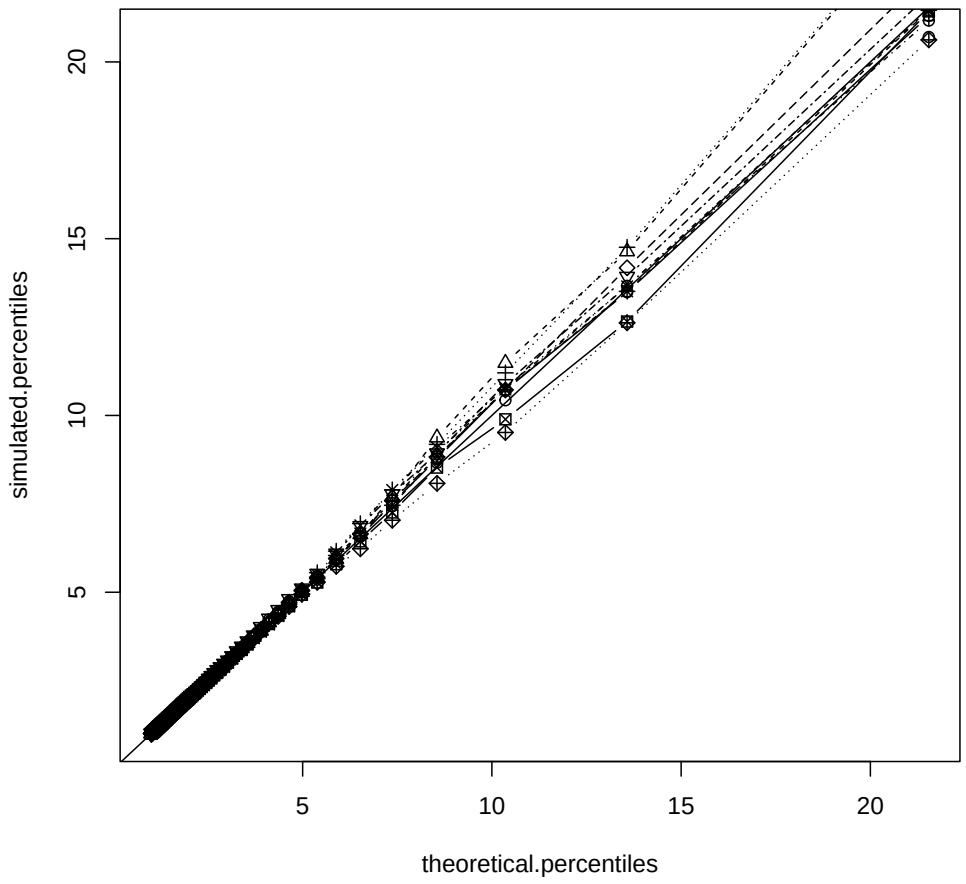
```
# Check Pareto random number generator, by repeatedly generating random draws
  # and comparing them to theoretical quantiles
# Inputs: Number of random points to generate per replication (n)
  # exponent of distribution (exponent)
  # lower threshold of distribution (threshold)
  # number of replications to create (B)
# Outputs: None
```




```
simulated.percentiles <- quantile(r,(0:99)/100)
theoretical.percentiles <- qpareto.4((0:99)/100,exponent=2.5,threshold=1)
plot(theoretical.percentiles,simulated.percentiles)
abline(0,1)
for (i in 1:10) { pareto.sim.vs.theory() }
```

Figure J.2 Comparing multiple simulated quantile values to the theoretical quantiles.

```
# Side-effects: Creates new plot, plots simulated points vs. theory
check.rpareto <- function(n=1e4, exponent=2.5, threshold=1, B=10) {
  # One set of percentiles for everything
  theoretical.percentiles <- qpareto.4((0:99)/100, exponent=exponent,
                                       threshold=threshold)
```




```
simulated.percentiles <- quantile(r,(0:99)/100)
theoretical.percentiles <- qpareto.4((0:99)/100,exponent=2.5,threshold=1)
plot(theoretical.percentiles,simulated.percentiles)
abline(0,1)
for (i in 1:10) {
  pareto.sim.vs.theory(pch=i,type="b",lty=i)
}
```

Figure J.3 As Figure J.2, but using the ability to pass along arguments to a subsidiary function to distinguish separate simulation runs.

```
  # Set up plotting window, but don't put anything in it:
  plot(0,type="n", xlim=c(0, max(theoretical.percentiles)),
       # No more horizontal room than we need
```

```
      ylim=c(0,1.1*max(theoretical.percentiles)),
      # Allow some extra vertical room for noise
      xlab="theoretical percentiles", ylab="simulated percentiles",
      main = paste("exponent = ", exponent, ", threshold = ", threshold))
  # Diagonal, for visual reference
  abline(0,1)
  for (i in 1:B) {
    pareto.sim.vs.theory(n=n, exponent=exponent, threshold=threshold,
                         pch=i, type="b", lty=i)
  }
}
```

R will accept this definition, but it won't run properly until we re-defined `pareto.sim.vs.theory` to take the arguments `n`, `exponent` and `threshold`.[5]

It seems like a simple modification of the old definition should do the trick:

```
# Compare random draws from Pareto distribution to theoretical quantiles
# Inputs: Graphical arguments, passed to points()
# Outputs: None
# Side-effects: Adds points showing random draws vs. theoretical quantiles
  # to current plot
pareto.sim.vs.theory <- function(n, exponent, threshold,...) {
  r <- rpareto(n=n, exponent=exponent, threshold=threshold)
  simulated.percentiles <- quantile(r, (0:99)/100)
  points(theoretical.percentiles, simulated.percentiles, ...)
}
```

After defining this, the checker function seems to work fine. The following commands produce the plot in Figure J.4, which looks very like the manually-created one. (Random noise means it won't be exactly the same.) Putting in the default arguments explicitly gives the same results (not shown).

```
check.rpareto()
check.rpareto(n=1e4, exponent=2.5, threshold=1)
```

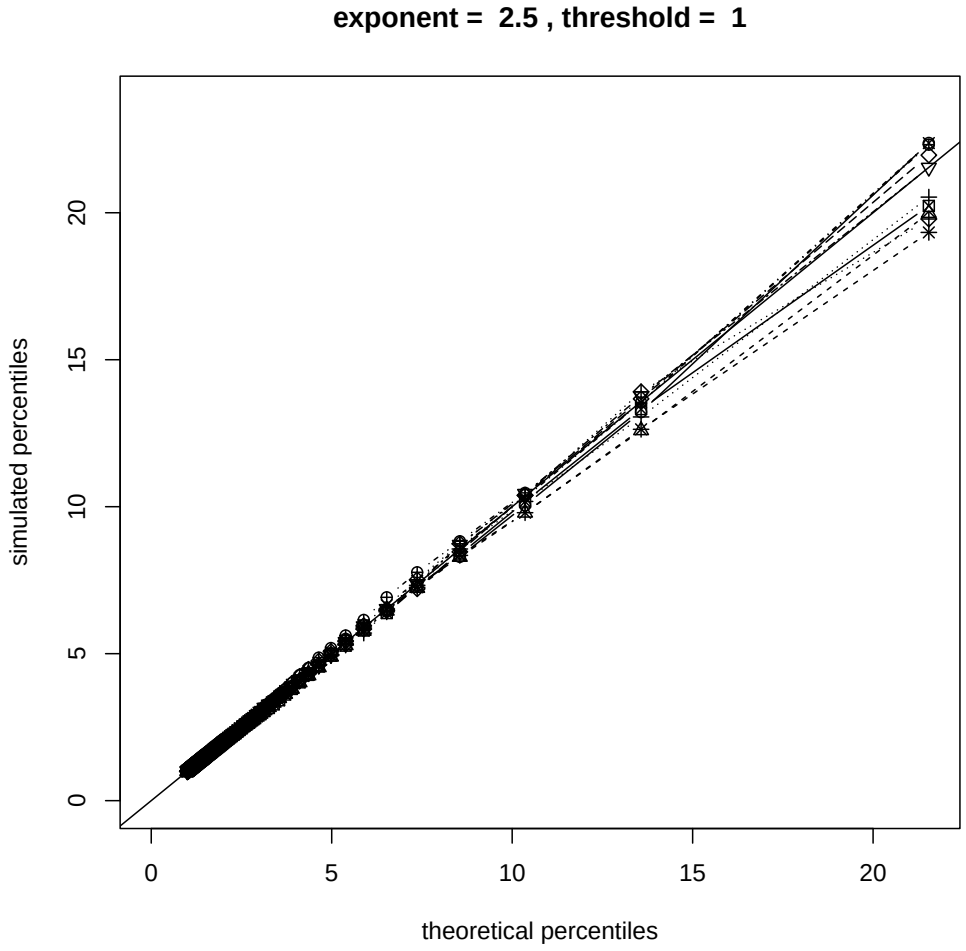
Unfortunately, changing the arguments reveals a bug (Figure J.5). Notice that the vertical coordinates of the points, coming from the simulation, look like they have about the same range as the theoretical quantiles, used to lay out the plotting window. But the horizontal coordinates are all pretty much the same (on a scale of tens of billions, anyway). What's going on?

The horizontal coordinates for the points being plotted are set in `pareto.sim.vs.theory.3`:

```
points(theoretical.percentiles, simulated.percentiles, ...)
```

Where does this function get `theoretical.percentiles` from? Since the variable isn't assigned inside the function, R tries to figure it out from context. Since `pareto.sim.vs.theory` was defined on the command line, the context R uses to interpret it is the global workspace — where there is, in fact, a variable called `theoretical.percentiles`, which I set by hand for the previous plots. So the

[5] Try running `check.rpareto()`, followed by `warnings()`.



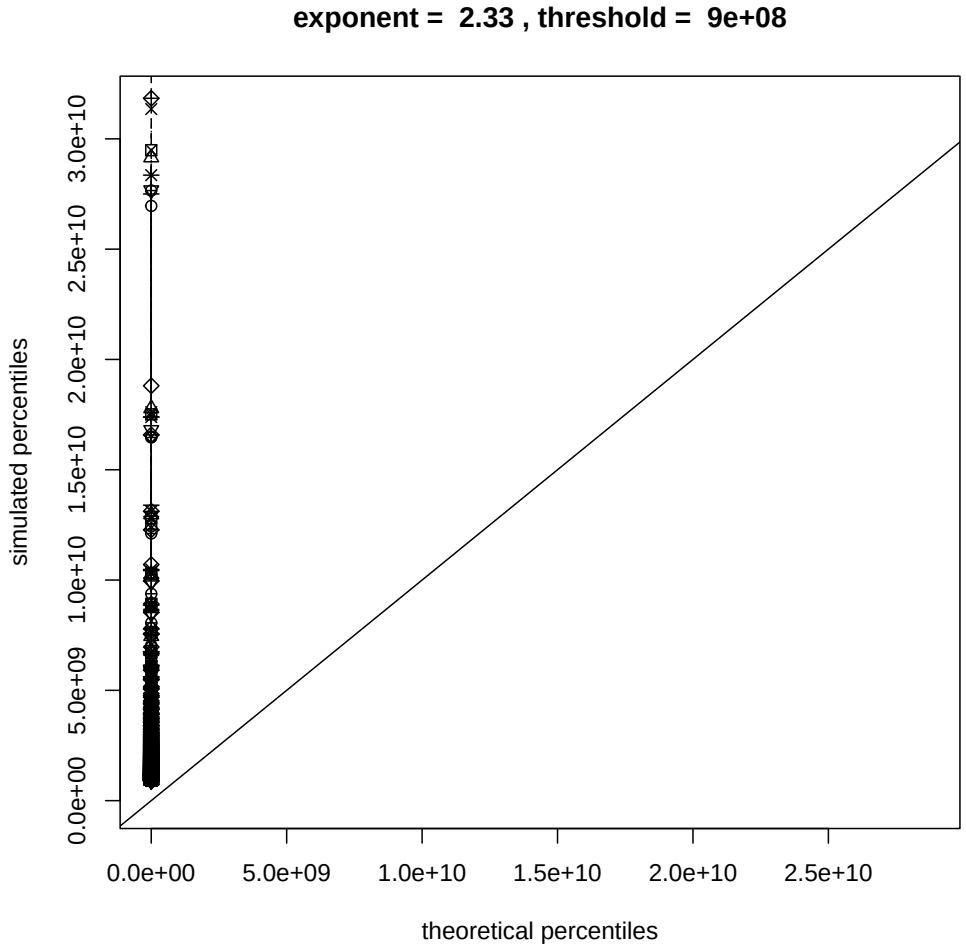

```
check.rpareto()
```

Figure J.4 Automating the checking of `rpareto`.

plotted theoretical quantiles are all too small in Figure J.5, because they're for a distribution with a much lower threshold.

Didn't `check.rpareto` assign is own value to `theoretical.percentiles`, which it used to set the plot boundaries? Yes, but that assignment only applied *in the context of the function*. Assignments inside a function have limited **scope**, they leave values in the broader context alone. Try this:

```
x <- 7
x
```




```
check.rpareto(n=1e4, exponent=2.33, threshold=9e8)
```

Figure J.5 A bug in `check.rpareto`.

```
## [1] 7
square <- function(y) { x <- y^2; return(x) }
square(7)
## [1] 49
x
## [1] 7
```

The function `square` assigns `x` to be the square of its argument. This assignment holds within the scope of the function, as we can see from the fact that the returned value is always the square of the argument, and not what we assigned

`x` to be in the global, command-line context. However, this does not over-write that global value, as the last line shows.[6]

There are two ways to fix this problem. One is to re-define `pareto.sim.vs.theory` to calculate the theoretical quantiles:

```
# Compare random draws from Pareto distribution to theoretical quantiles
# Inputs: Number of random points to generate (n)
  # exponent of distribution (exponent)
  # lower threshold of distribution (threshold)
  # graphical arguments, passed to points() (...)
# Outputs: None
# Side-effects: Adds points showing random draws vs. theoretical quantiles
  # to current plot
pareto.sim.vs.theory <- function(n, exponent, threshold,...) {
  r <- rpareto(n=n, exponent=exponent, threshold=threshold)
  theoretical.percentiles <- qpareto.4((0:99)/100, exponent=exponent,
                                       threshold=threshold)
  simulated.percentiles <- quantile(r,(0:99)/100)
  points(theoretical.percentiles, simulated.percentiles, ...)
}
```

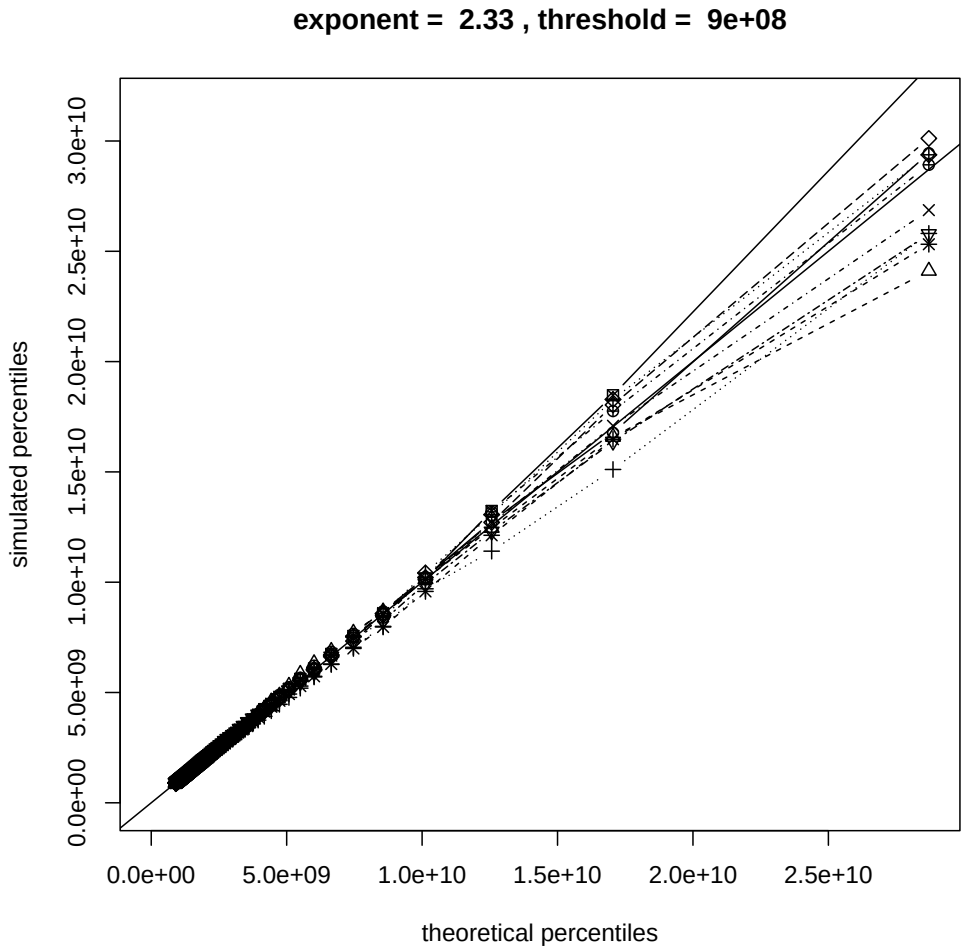
This will work (try running `check.rpareto(1e4,2.33,9e8)` now), but it's very redundant — every time we call this, we're recalculating the same percentiles, which we already calculated in `check.rpareto`. A cleaner solution is to make the vector of theoretical percentiles an argument to `pareto.sim.vs.theory`, and change `check.rpareto` to provide it.

```
# Compare random draws from Pareto distribution to theoretical quantiles
# Inputs: Graphical arguments, passed to points()
# Outputs: None
# Side-effects: Adds points showing random draws vs. theoretical quantiles
  # to current plot
check.rpareto <- function(n=1e4,exponent=2.5,threshold=1,B=10) {
  # One set of percentiles for everything
  theoretical.percentiles <- qpareto.4((0:99)/100,exponent=exponent,
    threshold=threshold)
  # Set up plotting window, but don't put anything in it:
  plot(0,type="n", xlim=c(0,max(theoretical.percentiles)),
    # No more horizontal room than we need
    ylim=c(0,1.1*max(theoretical.percentiles)),
    # Allow some extra vertical room for noise
    xlab="theoretical percentiles", ylab="simulated percentiles",
    main = paste("exponent = ", exponent, ", threshold = ", threshold))
  # Diagonal, for visual reference
  abline(0,1)
  for (i in 1:B) {
    pareto.sim.vs.theory(n=n,exponent=exponent,threshold=threshold,
      theoretical.percentiles=theoretical.percentiles,
      pch=i,type="b",lty=i)
  }
}
```

[6] There are techniques by which functions can change assignments outside of their scope. They are tricky, rare, and best avoided except by those who really know what they are doing. (If you think you do, you are probably wrong.)

```
# Compare random draws from Pareto distribution to theoretical quantiles
# Inputs: Number of random points to generate (n)
  # exponent of distribution (exponent)
  # lower threshold of distribution (threshold)
  # vector of theoretical percentiles (theoretical.percentiles)
  # graphical arguments, passed to points()
# Outputs: None
# Side-effects: Adds points showing random draws vs. theoretical quantiles
  # to current plot
pareto.sim.vs.theory <- function(n,exponent,threshold,
  theoretical.percentiles,...) {
  r <- rpareto(n=n,exponent=exponent,threshold=threshold)
  simulated.percentiles <- quantile(r,(0:99)/100)
  points(theoretical.percentiles,simulated.percentiles,...)
}
```

Figure J.6 shows that this succeeds.




```
check.rpareto(1e4,2.33,9e8)
```

Figure J.6 Using the corrected simulation checker.

J.6 Avoiding Iteration: Manipulating Objects

Let's go back to the declaration of `rpareto`, which I repeat here, unchanged, for convenience:

```
# Generate random numbers from the Pareto distribution
# Inputs: number of random draws (n)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
# Outputs: vector of random numbers
rpareto <- function(n,exponent,threshold) {
  x <- vector(length=n)
  for (i in 1:n) {
    x[i] <- qpareto.4(p=runif(1),exponent=exponent,threshold=threshold)
  }
  return(x)
}
```

We've confirmed that this works, but it involves explicit iteration in the form of the `for` loop. Because of the way R carries out iteration[7], it is slow, and better avoided when possible. Many of the utility functions in R, like `replicate`, are designed to avoid explicit iteration. We could re-write `rpareto` using `replicate`, for example:

```
# Generate random numbers from the Pareto distribution
# Inputs: number of random draws (n)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
# Outputs: vector of random numbers
rpareto <- function(n,exponent,threshold) {
  x <- replicate(n,qpareto.4(p=runif(1),exponent=exponent,threshold=threshold))
  return(x)
}
```

(The outstanding use of `replicate` is when we want to repeat the same random experiment many times — there are examples in the notes for Chapters 5 and 6.)

An even clearer alternative makes use of the way R automatically **vectorizes** arithmetic:

```
# Generate random numbers from the Pareto distribution
# Inputs: number of random draws (n)
  # exponent of the distribution (exponent)
  # lower threshold of the distribution (threshold)
# Outputs: vector of random numbers
rpareto <- function(n,exponent,threshold) {
  x <- qpareto.4(p=runif(n),exponent=exponent,threshold=threshold)
  return(x)
}
```

This feeds `qpareto.4` a *vector* of quantiles `p`, of length `n`, which in turn gets passed along to `qpareto.1`, which finally tries to evaluate

[7] Roughly speaking, it ends up having to create and destroy a whole copy of everything which gets changed in the course of one pass around the iteration loop, which can involve lots of memory and time.

```
threshold*((1-p)^(-1/(exponent-1)))
```

With `p` being a vector, R hopes that `threshold` and `exponent` are also vectors, and of the same length, so that it evaluates this arithmetic expression component-wise. If `exponent` and `threshold` are shorter, it will "recycle" their values, in order, until it has vectors equal in length to `p`. In particular, if `exponent` and `threshold` have length 1, it will repeat both of them `length(p)` times, and then evaluate everything component by component. (See the "Introduction to R" manual for more on this "recycling rule".) The quantile functions we have defined inherit this ability to recycle, without any special work on our part. The final version of `rpareto` we have written is not only faster, it is clearer and easier to read. It focuses our attention on what is being done, and not on the mechanics of doing it.

J.6.1 `ifelse` and `which`

Sometimes we want to do different things to different parts of a vector (or larger structure) depending on its values. For instance, in robust regression one often replaces the squared error loss with what's called the Huber loss[8],

$$\psi(x) = \begin{cases} x^2 & \text{if } |x| \leq 1 \\ 2|x| - 1 & \text{if } |x| > 1 \end{cases} \tag{J.4}$$

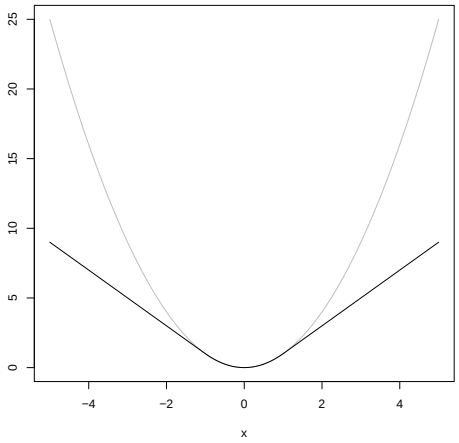
which isn't so vulnerable to outliers, as in Figure J.7.

We might code this up like so:

```
# Calculate Huber's loss function
# Input: vector of numbers x
# Return: x^2 for |x|<1, 2|x|-1 otherwise
huber <- function(x) {
  n <- length(x)
  y <- vector(n)
  for (i in 1:n) {
    if (abs(x) <= 1) {
      y[i] <- x[i]^2
    } else {
      y[i] <- 2*abs(x[i])-1
    }
  }
  return(y)
}
```

This is not very easy to follow. R provides a very useful function, `ifelse`, which lets us apply a binary test to each element in a vector, and then draw from either of two calculations. Using it, we re-write `huber` like so:

[8] One applies this not to the residuals directly, but to residuals divided by some robust measure of dispersion.




```
curve(x^2,col="grey",from=-5,to=5,ylab="")
curve(huber,add=TRUE)
```

Figure J.7 The Huber loss function ψ from Eq. J.4 (black) versus the squared error loss (grey).

```
# Calculate Huber's loss function
# Input: vector of numbers x
# Return: x^2 for |x|<1, 2|x|-1 otherwise
huber <- function(x) {
  return(ifelse(abs(x) <= 1, x^2, 2*abs(x)-1))
}
```

The first argument needs to produce a vector of TRUE/FALSE values; the second argument provides the outputs for the TRUE positions, the third outputs for the FALSE positions. Here all three are expressions involving the same variable, but that's not essential.

Another useful device is the `which` function, whose argument is a vector of TRUE/FALSE values, returning a vector of the indices where the argument is TRUE, e.g.,

```
incomplete.cases <- which(is.na(cholesterol))
```

would give us the positions at which the vector `cholesterol` had NA values. This is equivalent to

```
incomplete.cases <- c()
for (i in 1:length(cholesterol)) {
  if (is.na(cholesterol[i])) {
    incomplete.cases <- c(incomplete.cases,i)
  }
```

```
}
```

J.6.2 apply and Its Variants

Particularly useful ways of avoiding iteration come from the function `apply`, and the closely related `sapply` and `lapply` functions. (It particularly shows up `apply` in Chapter 6.)

```
x <- replicate(10,rpareto(100,2.5,1))
apply(x,2,quantile,probs=0.9)
```

Each call to `rpareto` inside the `replicate` creates a vector of length 100. Replicate then stacks these, as columns, into an array. The `apply` function applies the same function to each row or column of the array, depending on whether its second argument is 1 (rows) or 2 (columns). So this will find the 90th percentile of each of the 10 random-number draws, and give that back to us as a vector.

`array` only works for arrays, matrices and data frames (and works on them by treating them as arrays). If we want to apply the same function to every element of a vector or list, we use `lapply`. This gives us back a list, which can be inconvenient:

```
y <- c(0.9,0.99,0.999,0.99999)
lapply(y,qpareto.4,exponent=2.5,threshold=1)
## [[1]]
## [1] 4.641589
##
## [[2]]
## [1] 21.54435
##
## [[3]]
## [1] 100
##
## [[4]]
## [1] 2154.435
```

The function `sapply` works like `lapply`, but tries to simplify its output down to a vector or array:

```
sapply(y,qpareto.4,exponent=2.5,threshold=1)
## [1]    4.641589   21.544347  100.000000 2154.434690
```

That last line just did the equivalent of `qpareto.4(y,exponent=2.5,threshold=1)`, but `sapply` can take considerably more complicated functions:

```
# Suppose we have models lm.1 and lm.2 hanging around
some.models <- list(model.1=lm.1, model.2=lm.2)
# Extract all the coefficients from all the models
sapply(some.models,coefficients)
```

`sapply` has a `simplify` argument, which defaults to TRUE; setting it to `FALSE`

turns off the simplification. `replicate` has the same argument. Usually, simplifying the output of `sapply` or `replicate` is a good thing, but it can lead to weirdness when what's being repeated is a complicated value itself.

For instance, let's revisit the data set about economic growth and currency undervaluation across countries and times (Problem Set 2), and try fitting a different model for each five-year period.

```
uv <- read.csv("http://www.stat.cmu.edu/~cshalizi/uADA/16/hw/02/uv.csv")
uv.lm.fiveyear <- function(fiveyear) {
  lm(growth ~ log(gdp) + underval,data=uv[uv$year==fiveyear,])
}
# What are all the five-year periods in the data?
fiveyears <- sort(unique(uv$year))
fiveyear.models.1 <- sapply(fiveyears, uv.lm.fiveyear)
```

Working with `fiveyear.models.1` is going to be very hard, because it wants to be an array, but isn't quite, and is generally very confused. (Try it!) Instead do it this way:

```
fiveyear.models.2 <- sapply(fiveyears, uv.lm.fiveyear, simplify=FALSE)
```

`fiveyear.models.2` is simply a list with 10 elements, each one of which is an `lm`-style model. Now it's easy extract information about any particular one, or use `sapply`:

```
sapply(fiveyear.models.2, coefficients)
##                    [,1]         [,2]         [,3]         [,4]         [,5]
## (Intercept) -0.04635778 -0.045134479 -0.040404844 -0.045820302 -0.022554554
## log(gdp)     0.00843245  0.008659137  0.008534740  0.009419719  0.005690720
## underval    -0.00738292  0.003926747 -0.007497302 -0.007846092  0.004461034
##                    [,6]         [,7]        [,8]         [,9]        [,10]
## (Intercept) -0.011886137 -0.028066634 -0.10547596 -0.038967138 -0.054008775
## log(gdp)     0.002667598  0.004361408  0.01358393  0.006042791  0.008512894
## underval     0.013164665  0.007724422  0.01808939 -0.011033117  0.019044209
```

J.7 More Complicated Return Values

So far, all the functions we have written have returned either a single value, or a simple vector, or nothing at all. The built-in functions return much more complicated things, like matrices, data frames, or lists, and we can too.

To illustrate, let's switch gears away from the Pareto distribution, and think about the Gaussian for a change. As you know, if we have data $x_1, x_2, \ldots x_n$ and we want to fit a Gaussian distribution to them by maximizing the likelihood, the best-fitting Gaussian has mean

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{J.5}$$

which is just the sample mean, and variance

$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2 \tag{J.6}$$

which differs from the usual way of defining the sample variance by having a factor of n in the denominator, instead of $n-1$. Let's write a function which takes in a vector of data points and returns the maximum-likelihood parameter estimates for a Gaussian.

```
gaussian.mle <- function(x) {
  n <- length(x)
  mean.est <- mean(x)
  var.est <- var(x)*(n-1)/n
  est <- list(mean=mean.est, sd=sqrt(var.est))
  return(est)
}
```

There is one argument, which is the vector of data. To be cautious, I should probably check that it *is* a vector of numbers, but skip that to be clear here. The first line figures out how many data points we have. The second takes the mean. The third finds the estimated variance — the definition of the built-in `var` function uses $n-1$ in its denominator, so I scale it down by the appropriate factor[9]. The fourth line creates a list, called `est`, with two components, named `mean` and `sd`, since those are the names R likes to use for the parameters of Gaussians. The first component is our estimated mean, and the second is the standard deviation corresponding to our estimated variance[10]. Finally, the function returns the list.

As always, it's a good idea to check the function on a case where we know the answer.

```
x <- 1:10
mean(x)
## [1] 5.5
var(x) * (length(x)-1)/length(x)
## [1] 8.25
sqrt(var(x) * (length(x)-1)/length(x))
## [1] 2.872281
gaussian.mle(x)
## $mean
## [1] 5.5
##
## $sd
## [1] 2.872281
```

[9] Clearly, if n is large, $\frac{n-1}{n} = 1 - 1/n$ will be very close to one, but why not be precise?

[10] If n is large, $\sqrt{\frac{n-1}{n}} = \sqrt{1-\frac{1}{n}} \approx 1 - \frac{1}{2n}$ (using the binomial theorem in the last step). For reasonable data sets, the error of just using `sd(x)` would have been small — but why have it at all?

J.8 Re-Writing Your Code: An Extended Example

Suppose we want to find a standard error for the median of a Gaussian distribution. We know, somehow, that the mean of the Gaussian is 3, the standard deviation is 2, and the sample size is one hundred. If we do

```
x <- rnorm(n=100,mean=3,sd=2)
```

we'll get a draw from that distribution in `x`. If we do

```
x <- rnorm(n=100,mean=3,sd=2)
median(x)
## [1] 2.877468
```

we'll calculate the median on one random draw. Following the general idea of Monte Carlo (§5.4.1) we can approximate the standard error of the median by repeating this calculation many times, on many random draws, and taking the standard deviation. We'll do this by explicitly iterating, so we need to set up a vector to store our intermediate results first.

```
medians <- vector(length=100)
for (i in 1:100) {
  x <- rnorm(n=100,mean=3,sd=2)
  medians[i] <- median(x)
}
se.in.median <- sd(medians)
```

Well, how do we know that 100 replicates is enough to get a good approximation? We'd need to run this a couple of times, typing it in or at least pasting it in many times. Instead, we can write a function which just gives everything we've done a single name. (I'll add comments as I go on.)

```
# Inputs: None (everything is hard-coded)
# Output: the standard error in the median
find.se.in.median <- function() {
  # Set up a vector to store the simulated medians
  medians <- vector(length=100)
  # Do the simulation 100 times
  for (i in 1:100) {
    x <- rnorm(n=100,mean=3,sd=2) # Simulate
    medians[i] <- median(x) # Calculate the median of the simulation
  }
  se.in.median <- sd(medians) # Take standard deviation
  return(se.in.median)
}
```

If we decide that 100 replicates isn't enough and we want 1000, we need to change this function. We could just change the first two appearances of "100" to "1000", but we have to catch all of them; we have to remember that the 100 in `rnorm` is there for a different reason and leave it alone; and if we later decide that actually 500 replicates would be enough, we have to do everything all over again.

It is easier, safer, clearer and more flexible to abstract a little and add an

argument to the function, which is the number of replicates. I'll add comments as I go.

```
# Inputs: Number of replicates (B)
# Output: the standard error in the median
find.se.in.median <- function(B) {
  # Set up a vector to store the simulated medians
  medians <- vector(length=B)
  # Do the simulation B times
  for (i in 1:B) {
    x <- rnorm(n=100,mean=3,sd=2) # Simulate
    medians[i] <- median(x) # Calculate median of the simulation
  }
  se.in.median <- sd(medians) # Take standard deviation
  return(se.in.median)
}
```

Now suppose we want to find the standard error of the median for an exponential distribution with rate 2 and sample size 37. We could write another function,

```
find.se.in.median.exp <- function(B) {
  # Set up a vector to store the simulated medians
  medians <- vector(length=B)
  # Do the simulation B times
  for (i in 1:B) {
    x <- rexp(n=37,rate=2) # Simulate
    medians[i] <- median(x) # Calculate median of the simulation
  }
  se.in.median <- sd(medians) # Take standard deviation
  return(se.in.median)
}
```

but it is wasteful to define two functions which do almost the same job. It's not just inelegant; it invites mistakes, it's harder to read (imagine coming back to this in two weeks — was there a big reason why we had two separate functions here?), and it's harder to improve. We need to abstract a bit more.

We *could* put in some kind of switch which would simulate from either of these two distributions, maybe like this:

```
# Inputs: number of replicates (B)
  # flag for whether to use a normal or an exponential (use.norm)
# Output: The standard error in the median
find.se.in.median <- function(B,use.norm=TRUE) {
  medians <- vector(length=B)
  for (i in 1:B) {
    if (use.norm) {
      x <- rnorm(100,3,2)
    } else {
      x <- rexp(37,2)
    }
    medians[i] <- median(x)
  }
  se.in.median <- sd(medians)
  return(se.in.median)
```

```
}
```

But why just these two? If we wanted any other distribution whatsoever, plainly all we'd have to do is change how `x` is simulated. So we really want to be able to *give* a simulator to the median-finding function as an argument.

Fortunately, in R you can give one function as an argument to another, so we'd do something like this.

```
# Inputs: Number of replicates (B)
  # Simulator function (simulator)
# Presumes: simulator is a no-argument function which produce a vector of
  # numbers
# Output: The standard error in the media
find.se.in.median <- function(B,simulator) {
  median <- vector(length=B)
  for (i in 1:B) {
    x <- simulator()
    medians[i] <- median(x)
  }
  se.in.medians <- sd(medians)
  return(se.in.medians)
}
```

Now to repeat our original calculations, we define a simulator function:

```
# Inputs: None
# Output: ten draws from the mean 3, s.d. 2 Gaussian
simulator.1 <- function() {
  return(rnorm(100,3,2))
}
```

If we now call this function, then every time `find.se.in.median` goes through the `for` loop, it will call `simulator.1`, which in turn will produce the right random numbers.

```
find.se.in.median(B=100,simulator=simulator.1)
## [1] 0.2649333
```

If we also define

```
# Inputs: None
# Output: 37 draws from the rate 2 exponential
simulator.2 <- function() {
  return(rexp(37,2))
}
```

then to find the standard error in the median of *this*, we just call

```
find.se.in.median(B=100, simulator=simulator.2)
## [1] 0.09539154
```

This same approach works if we want to sample from a much more complicated distribution. If we fit a kernel regression to the data on economic growth and currency undervaluation (Problem Set 2), and want a standard error in the

median of the predicted growth rate, with noise coming from resampling cases, we would do something like this for the simulator

```
# Perturb the currency-undervaluation data by re-sampling and fit a kernel
  # regression for growht on initial GDP and undervaluation
# Inputs: None
# Output: The fitted growth rates from a new kernel regression
simulator.3 <- function() {
  # Make sure the np library is loaded
  require(np)
  # If we haven't already loaded the data, load it
  if (!exists("uv")) {
    uv <- read.csv("http://www.stat.cmu.edu/~cshalizi/uADA/16/hw/02/uv.csv")
  }
  # How big is the data set?
  n <- nrow(uv)
  # Treat the data set like a population and draw a sample
  resampled.rows <- sample(1:n,size=n,replace=TRUE)
  uv.r <- uv[resampled.rows,]
  # See the chapter on smoothing for the following incantation
  fit <- npreg(growth~log(gdp)+underval, data=uv.r, tol=1e-2, ftol=1e-2)
  growth.rates <- fitted(fit)
  return(growth.rates)
}
```

[[TODO: Increase number of replicates for production draft]]

and then this to find the standard error in the median:

```
find.se.in.median(B=10, simulator=simulator.3)
## [1] 0.9299164
```

By breaking up the task this way, if we encounter errors or just general trouble when we run that last command, it is easier to localize the problem. We can check whether `find.se.in.median` seems to work properly with other simulator functions. (For instance, we might write a "simulator" that either does `rep(10,1)` or `rep(10,-1)` with equal probability, since then we can work out what the standard error of the median ought to be.) We can also check whether `simulator.3` is working properly, and finally whether there is some issue with putting them together, say that the output from the simulator is not quite in a format that `find.se.in.median` can handle. If we just have one big ball of code, it is much harder to read, to understand, to debug, and to improve.

To turn to that last point, one of the things R does poorly is explicit iteration with `for` loops. As mentioned in §J.6, it's generally better to replace such loops with "vectorized" functions, which do the iteration using fast code outside of R. One of these, especially for this situation, is the function `replicate`. We can re-write `find.se.in.median` using it:

```
# Inputs: number of replicates (B)
  # Simulator function (simulator)
# Presumes: simulator is a no-argument function which produces a vector of
  # numbers
# Outputs: Standard error in the median of the output of simulator
find.se.in.median <- function(B,simulator) {
  medians <- replicate(B, median(simulator()))
```

```
  se.in.median <- sd(medians)
  return(se.in.median)
}
```

Again: shorter, faster, and easier to understand (if you know what `replicate` does). Also, because we are telling this what simulation function to use, and writing those functions separately, we do not have to change any of our simulators. They don't *care* how `find.se.in.median` works. In fact, they don't care that there is any such function — they could be used as components in many other functions which can also process their outputs. So long as these *interfaces* are maintained, the inner workings of the functions are irrelevant to each other.

Suppose for instance that we want not the standard error of the median, but the interquartile range of the median — the median is after all a "robust", outlier-resistant measure of the central tendency, and the IQR is likewise a robust measure of dispersion. This is now easy:

```
# Inputs: number of replicates (B)
  # Simulator function (simulator)
# Presumes: simulator is a no-argument function which produces a vector of
  # numbers
# Outputs: Interquartile range of the median of the output of simulator
find.iqr.of.median <- function(B,simulator) {
  medians <- replicate(B,median(simulator()))
  iqr.of.median <- IQR(medians)
  return(iqr.of.median)
}
```

Or for that matter the good old standard error of the mean:

```
# Inputs: number of replicates (B)
  # Simulator function (simulator)
# Presumes: simulator is a no-argument function which produces a vector of
  # numbers
# Outputs: Standard error of the mean of the output of simulator
find.se.of.mean <- function(B,simulator) {
  means <- replicate(B,mean(simulator()))
  se.of.mean <- sd(means)
  return(se.of.mean)
}
```

These last few examples suggest that we could abstract even further, by swapping in and out different estimators (like `median` and `mean`) and different summarizing functions (like `se` or `IQR`).

```
# Inputs: number of replicates (B)
  # Simulator function (simulator)
  # Estimator function (estimator)
  # Sample summarizer function (summarizer)
# Presumes: simulator is a no-argument function which produces a vector of
  # numbers
  # estimator is a function that takes a vector of numbers and produces one
  # output
  # summarizer takes a vector of outputs from estimator
```

```
# Outputs: Summary of the simulated distribution of estimates
summarize.sampling.dist.of.estimates <- function(B,simulator,estimator,
                                                 summarizer) {
  estimates <- replicate(B,estimator(simulator()))
  return(summarizer(estimates))
}
```

The name is too long, of course, so we should replace it with something catchier (Chapter 6):

```
bootstrap <- function(B,simulator,estimator,summarizer) {
  estimates <- replicate(B,estimator(simulator()))
  return(summarizer(estimates))
}
```

Our very first example in this section is equivalent to

```
bootstrap(B=100,simulator=simulator.1, estimator=median, summarizer=sd)
## [1] 0.232511
```

`bootstrap` is just two lines: one simulates and re-estimates, the other summarizes the re-estimates. This is the essence of what we are trying to do, and is logically distinct from the details of particular simulators, estimators and summaries.

We started with a particular special case and generalized it. The alternative route is to start with a very general framework — here, writing `bootstrap` — and then figure out what lower-level functions we would need to make it work in a the case at hand, writing them if necessary. (We need to write a simulator, but someone's already written `median` for us.) Getting the first stage right involves a certain amount of reflection on how to solve the problem — it's rather like doing a "show that" math problem by starting from the desired conclusion and working backwards.

It is still somewhat clunky to have to write a new function every time we want to change the settings in the simulation, but this has gone on long enough.

J.9 General Advice on Programming

Programming is an act of communication: with the computer, of course, but also with your co-workers, and with yourself in the future[11]. Clear and effective communication is a valuable skill in itself; it also tends to make it easier to do the job, and to make debugging easier. This section, then, gives some general advice about making your programs clearer and more effective, closing (in §J.9.7) by going over how I used these principles when writing code to implement simulation-based estimation for a time-series model in Chapter 24.

[11] And, in a class, with your graders.

J.9.1 Comment your code

Comments lengthen your file, but they make it immensely easier for other people to understand. ("Other people" includes your future self; there are few experiences more frustrating than coming back to a program after a break only to wonder what you were thinking.) Comments should say what each part of the code does, and how it does it. The "what" is more important; you can change the "how" more often and more easily.

Every function (or subroutine, etc.) should have comments at the beginning saying:

- what it does;
- what all its inputs are (in order);
- what it requires of the inputs and the state of the system ("presumes");
- what side-effects it may have (e.g., "plots histogram of residuals");
- what all its outputs are (in order)

Listing what other functions or routines the function calls ("dependencies") is optional; this can be useful, but it's easy to let it get out of date.

You should treat "Thou shalt comment thy code" as a commandment which Moses brought down from Mt. Sinai, written on stone by a fiery Hand.

J.9.2 Use meaningful names

Unlike some older languages, R lets you give variables and functions names of essentially arbitrary length and form. So give them meaningful names. Writing `loglikelihood`, or even `loglike`, instead of `L` makes your code a little longer, but generally a lot clearer, and it runs just the same.

This rule is lower down in the list because there are exceptions and qualifications. If your code is tightly associated to a mathematical paper, or to a field where certain symbols are conventionally bound to certain variables, you may as well use those names (e.g., call the probability of success in a binomial `p`). You should, however, explain what those symbols are in your comments. In fact, since what you regard as a meaningful name may be obscure to others (e.g., those grading your work), you should use comments to explain variables in any case. Finally, it's OK to use single-letter variable names for counters in loops (but see the advice on iteration in §J.6).

J.9.3 Check whether your program works

It's not enough — in fact it's very little — to have a program which runs and gives you some output. It needs to be the right output. You should therefore construct tests, which are things that the correct program should be able to do, but an incorrect program should not. This means that:

- you need to be able to check whether the output is right;

- your tests should be reasonably severe, so that it's hard for an incorrect program to pass them;
- your tests should help you figure out what isn't working;
- you should think hard about programming the test, so it checks whether the output is right, and you can easily repeat the test as many times as you need.

Try to write tests for the component functions, as well as the program as a whole. That way you can see where failures are. Also, it's easier to figure out what the right answers should be for small parts of the problem than the whole.

Try to write tests as very small functions which call the component you're testing with controlled input values. For instance, we tested `qpareto` by looking at what it returned for selected arguments with manually carrying out the computation. With statistical procedures, tests can look at average or distributional results — we saw an example of this with checking `rpareto`.

Of course, unless you are very clever, or the problem is very simple, a program could pass all your tests and still be wrong, but a program which fails your tests is definitely not right.

(Some people would actually advise writing your tests before writing any actual functions. They have a point, but I think that's overkill for this class.)

J.9.4 Avoid writing the same thing twice

Many data-analysis tasks involve doing the same thing multiple times, either as iteration, or to slightly different pieces of data, or with some parameters adjusted, etc. Try to avoid writing two pieces of code to do the same job. If you find yourself copying the same piece of code into two places in your program, look into writing *one* function, and *calling* it twice.

Doing this means that there is only one place to make a mistake, rather than many. It also means that when you fix your mistake, you only have one piece of code to correct, rather than many. (Even if you don't make a mistake, you can always make improvements, and then there's only one piece of code you have to work on.) It also leads to shorter, more comprehensible and more adaptable code.

J.9.5 Start from the beginning and break it down

When you have a big problem, start by thinking about what you want your program to do. Then figure out a set of slightly smaller steps which, put together, would accomplish that. Then take each of those steps and break them down into yet smaller ones. Keep going until the pieces you're left with are so small that you can see how to do each of them with only a few lines of code. Then write the code for the smallest bits, check it, once it works write the code for the next larger bits, and so on.

In slogan form:

- Think before you write.

- What first, then how.
- Design from the top down, code from the bottom up.

(Not everyone likes to design code this way, and it's not in the written-in-stone-atop-Sinai category, but there are many much worse ways to start.)

J.9.6 Break your code into many short, meaningful functions

Since you have broken your programming problem into many small pieces, try to make each piece a short function. (In other languages you might make them subroutines or methods, but in R they should be functions.)

Each function should achieve a single coherent task — its function, if you will. The division of code into functions should respect this division of the problem into sub-problems. More exactly, the way you break your code into functions is how you have divided your problem.

Each function should be short, generally less than a page of print-out. The function should do one single meaningful thing. (Do not just break the calculation into arbitrary thirty-line chunks and call each one a function.) These functions should generally be separate, not nested one inside the other.

Using functions has many advantages:

- you can re-use the same code many times, either at different places in this program or in other programs
- the rest of your code only has to care about the inputs and outputs to the function (its interfaces), not about the internal machinery that turns inputs into outputs. This makes it easier to design the rest of the program, and it means you can change that machinery without having to re-design the rest of the program.
- it makes your code easier to test (see below), to debug, and to understand.

Of course, every function should be commented, as described above.

J.9.7 Illustration: The Method of Moments Code from §24.1.3

This section goes over the code for the method of moments in §24.1.3 as an example of how to write code in R, using the principles above.

The first function, `ma.mm.est`, estimates the parameters taking as inputs two numbers, representing the covariance and the variance. The real work is done by the built-in `optim` function[12], which itself takes two major arguments. One, `fn`, is the function to optimize. Another, `par`, is an initial guess about the parameters at which to begin the search for the optimum.[13]

The `fn` argument to `optim` must be a function, here `ma.mm.objective`. The first argument to that function has to be a vector, containing all the parameters

[12] See §D.4.

[13] Here `par` is a very rough guess based on `c` and `v` — it'll actually be right when `c`=0, but otherwise it's not much good. Fortunately, it doesn't have to be! Anyway, let's return to designing the code

to be optimized over. (Otherwise, `optim` will quit and complain.) There can be other arguments, not being optimized over, to that function, which `optim` will pass along, as you see here. `optim` will also accept a lot of optional arguments to control the search for the optimum — see `help(optim)`.

All `ma.mm.objective` has to do is calculate the objective function. The first two lines peel out θ and σ^2 from the parameter vector, just to make it more readable. The next two lines calculate what the moments should be. The last line calculates the distance between the model predicted moments and the actual ones, and returns it. The whole thing could be turned into a one-liner, like

```
return(t(params-c(c,v)) %*% (params-c(c,v)))
```

or perhaps even more obscure, but that is usually a bad idea.

Notice that I could write these two functions independently of one another, at least to some degree. When writing `ma.mm.est`, I knew I would need the objective function, but all I needed to know about it was its name, and the promise that it would take a parameter vector and give back a real number. When writing `ma.mm.objective`, all I had to remember about the other function was the promise this one needed to fulfill. In my experience, it is usually easiest to do any substantial coding in this "top-down" fashion[14]. Start with the high-level goal you are trying to achieve, break it down into a few steps, write something which will put those steps together, presuming other functions or programs can do them. Now go and write the functions to do each of those steps.

The code for the method of simulated moments is entirely parallel to these. Writing it as two separate pairs of functions is therefore somewhat wasteful. If I find a mistake in one pair, or thing of a way to improve it, I need to remember to make corresponding changes in the other pair (and not introduce a new mistake). In the long run, when you find yourself writing parallel pieces of code over and over, it is better to try to pull together the common parts and write them *once*. Here, that would mean something like one pair of functions, with the inner one having an argument which controlled whether to calculate the predicted moments by simulation or by a formula. You may try your hand at writing this.

J.10 Further Reading

Matloff (2011) is a good introduction to programming for total novices using R. Braun and Murdoch (2008) has more on statistical calculations and related topics, but can also work as an introduction for absolute beginners. Adler (2009) is an introduction to R for those with some prior knowledge of other programming languages. For sheer data manipulation, see Spector (2008). Chambers (2008) and Wickham (2015) are both essential for anyone who wants to be serious about programming in R.

If you are going to do a lot of computational work, it is worthwhile learning some of what programmers are taught. The "Software Carpentry" website (`http://software-carpentry.org`) provides good introduction to key tools, like the

[14] What qualifies as "substantial coding" depends on how much experience you have

Unix shell and version control. It is also worth learning about common data structures and the algorithms for working with them, since the right choices there can make dramatic differences; I like Cormen *et al.* (2001), but there are many fine alternatives.

Data-Analysis Assignments

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

[[TODO: Renew page numbers from 1 here]]

All of the following problem sets have been used in class at least once. They are arranged in an order approximately matching the order of the chapters, but many of them draw on multiple chapters. Each one is scored out of 90 points, with an extra 10 points allocated to clarity of writing, figures, code, etc. (The exact rubric is given below.) ; in a typical semester, students would do one problem set a week, 12–14 in all. A few provide much less "scaffolding" to guide students through the analysis; these were assigned as take-home exams.

[[TODO: Fix so they really are scored from 90]]

Most of these assignments are based on published papers in the scientific or statistical literature; I have provided citations to the source papers, but urge students to *not* read them until after they have attempted the assignment.[15]

[[TODO: Add references to the source papers]]

0.11 Suggested rubric for writing and formatting

This describes the ideal; the suggested weight is 10 points out of 100.

The text is laid out cleanly, with clear divisions between problems and subproblems. The writing itself is well-organized, free of grammatical and other mechanical errors, and easy to follow. Figures and tables are easy to read, with informative captions, axis labels and legends, and are placed near the text of the corresponding problems. All quantitative and mathematical claims are supported by appropriate derivations, included in the text, or calculations in code. Numerical results are reported to appropriate precision. Code is either properly integrated with a tool like R Markdown or knitr, or included as a separate R file. In the former case, both the knitted and the source file are included. In the latter case, the code is clearly divided into sections referring to particular problems. In either case, the code is indented, commented, and uses meaningful names. All code is relevant to the text; there are no dangling or useless commands. All parts of all problems are answered with actual coherent sentences, and never with raw computer code or its output. For full credit, all code runs, and the Markdown file knits (if applicable).

[15] Some of the source papers would be positive hindrances.

1

Your Daddy's Rich

When the assignment says "make a scatterplot of A against B", or "plot A against B", A goes on the vertical axis and B on the horizontal axis.

AGENDA: Getting back into practice with regression; starting to unlearn some bad habits.

Source: Chetty *et al.* (2014)

This assignment will look at economic mobility across generations in the contemporary USA. The data come from a large study, based on tax records, which allowed researchers to link the income of adults to the income of their parents several decades previously. For privacy reasons, we don't have that individual-level data, but we do have aggregate statistics about economic mobility for several hundred communities, containing most of the American population, and covariate information about those communities. We are interested in predicting economic mobility from the characteristics of communities.

The Data

The data file `mobility.csv` has information on 741 communities[1]. The variable we want to predict is economic mobility; the rest are predictor variables or covariates.

1. Mobility: The probability that a child born in 1980–1982 into the lowest quintile (20%) of household income will be in the top quintile at age 30. Individuals are assigned to the community they grew up in, not the one they were in as adults.
2. Population in 2000.
3. Is the community primarily urban or rural?
4. Black: percentage of individuals who marked black (and nothing else) on census forms.
5. Racial segregation: a measure of residential segregation by race.
6. Income segregation: Similarly but for income.
7. Segregation of poverty: Specifically a measure of residential segregation for those in the bottom quarter of the national income distribution.
8. Segregation of affluence: Residential segregation for those in the top qarter.
9. Commute: Fraction of workers with a commute of less than 15 minutes.
10. Mean income: Average income per capita in 2000.

[1] Technically, "commuting zones". These include cities and their suburbs and exurbs, but also many rural areas with integrated economies.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

11. Gini: A measure of income inequality, which would be 0 if all incomes were perfectly equal, and tends towards 100 as all the income is concentrated among the richest individuals (see Wikipedia, s.v. "Gini coefficient").
12. Share 1%: Share of the total income of a community going to its richest 1%.
13. Gini bottom 99%: Gini coefficient among the lower 99% of that community.
14. Fraction middle class: Fraction of parents whose income is between the *national* 25th and 75th percentiles.
15. Local tax rate: Fraction of all income going to local taxes.
16. Local government spending: per capita.
17. Progressivity: Measure of how much state income tax rates increase with income.
18. EITC: Measure of how much the state contributed to the Earned Income Tax Credit (a sort of negative income tax for very low-paid wage earners).
19. School expenditures: Average spending per pupil in public schools.
20. Student/teacher ratio: Number of students in public schools divided by number of teachers.
21. Test scores: *Residuals* from a linear regression of mean math and English test scores on household income per capita.
22. High school dropout rate: Also, *residuals* from a linear regression of the dropout rate on per-capita income.
23. Colleges per capita
24. College tuition: in-state, for full-time students
25. College graduation rate: Again, *residuals* from a linear regression of the actual graduation rate on household income per capita.
26. Labor force participation: Fraction of adults in the workforce.
27. Manufacturing: Fraction of workers in manufacturing.
28. Chinese imports: Growth rate in imports from China per worker between 1990 and 2000.
29. Teenage labor: fraction of those age 14–16 who were in the labor force.
30. Migration in: Migration into the community from elsewhere, as a fraction of 2000 population.
31. Migration out: Ditto for migration into other communities.
32. Foreign: fraction of residents born outside the US.
33. Social capital: Index combining voter turnout, participation in the census, and participation in community organizations.
34. Religious: Share of the population claiming to belong to an organized religious body.
35. Violent crime: Arrests per person per year for violent crimes.
36. Single motherhood: Number of single female households with children divided by the total number of households with children.
37. Divorced: Fraction of adults who are divorced.
38. Married: Ditto.
39. Longitude: Geographic coordinate for the center of the community
40. Latitude: Ditto
41. ID: A numerical code, identifying the community.

42. Name: the name of principal city or town.
43. State: the state of the principal city or town of the community.

Some of these variables are missing for some communities, and this may make a difference for some questions.

1. (5) Draw a map of mobility. That is, make a plot where the x and y coordinates are longitude and latitude, and mobility is indicated by color (possibly grey scale), by a third coordinate, or some other suitable device. Make sure your map is legible. Describe the geographic pattern in words.
2. (15) Make scatter plots of mobility against each of the following variables. Include on each plot a line for the simple or univariate regression, and give a table of the regression coefficients. Carefully explain the interpretation of each coefficient. (2 pts each) Do any of the results seem odd? (1 pt)
    1. Population
    2. Mean household income per capita
    3. Racial segregation
    4. Income share of the top 1%
    5. Mean school expenditures per pupil
    6. Violent crime rate
    7. Fraction of workers with short commutes.
3. Run a linear regression of mobility against all appropriate covariates.
    1. (5) Report all regression coefficients and their standard errors to *reasonable* precision; you may use either a table or a figure as you prefer. Do not just paste in R's output.
    2. (1) Explain why the ID variable must be excluded.
    3. (4) Explain which other variables, if any, you excluded from the regression, and why. (If you think they can all be used, explain why.)
    4. (5) Compare the coefficients you found in problem 2 to the coefficients for the same variables in this regression. Are they significantly different? Have any changed sign?
4. *The wrong side of the tracks starts at Giant Eagle* Find Pittsburgh in the data set.
    1. (1) What its actual mobility? What is its predicted mobility, according to the model?
    2. (3) Holding all else fixed, what is the predicted mobility if the violent crime rate is doubled? If it is halved?
    3. (3) Holding all else fixed, at what level of income segregation does the model predict that mobility will exceed 1.0?
    4. (3) Holding all else fixed, what would the income share of the top 1% have to be for the model to predict that mobility will fall to 0.0?

    (We will see later in the course how to avoid the embarrassment of models that predict probabilities greater than 1 or less than 0.)
5. *Free as in beer*
    1. (1) The national mobility level is the average mobility across all communities, weighted by population. What is it?
    2. (3) Suppose college were made free for everyone. Calculate the *change* in the predicted mobility for each community. Report the minimum, median, mean and maximum changes.

3. (1) Find the change to the predicted (not actual) national mobility level from making college free for everyone. *Hint:* consider a weighted average, or weighted sum, of your vector of answers from Problem 52.
4. (3) Give a (rough) 95% confidence interval for the change in the predicted national mobility level.
5. (2) Explain at least one way in which this calculation is unrealistic.

6. *Distinctions vs. differences*
    1. (2) Make a table ranking the variables by the magnitude of the t statistic in the regression results (i.e., rank by $|t|$, not t).
    2. (6) For each variable in the model, find the expected change in mobility from a one standard deviation change in that variable (assuming all else is fixed). Provide a table ranking variables by the magnitude of their impact.
    3. (2) How similar is the ranking by impact to the ranking by t statistics?
7. (5) Make a map of the model's predicted mobility. How does it compare, qualitatively, to the map of actual mobility?
8. *After making proper allowances*
    1. (1) Make a map of the model's residuals.
    2. (2) What are the five communities with the largest positive residuals? The five with the most negative residuals? (Can you mark these on the map?)
    3. (2) One interpretation of these residuals is that they show communities where some factor not included in the model leads to higher (or lower) mobility than in otherwise-similar communities. Suggest at least one other interpretation. Could you test these ideas with this data set?
9. *Expectations and reality*
    1. (3) Make a scatterplot of actual mobility against predicted mobility. Is the relationship linear? Should it be, if the model is right? Is the relationship flat? Should it be, if the model is right?
    2. (2) Make a scatterplot of the model's residuals against predicted mobility. Is the relationship linear? Should it be, if the model is right? Is the relationship flat? Should it be, if the model is right?
10. *Model checking will continue until morale improves*
    1. (5) For *each* variable in the model, make a scatterplot of the model's residuals against the predictor variable. (You will have a lot of plots.)
    2. (5) Explain why, if the linear model is right, all the relationships you just plotted should be flat.
    3. (5) Explain why, if the usual assumptions for t tests and their p-values are right, each plot should have a roughly constant vertical spread of points as one moves from left to right.
    4. (5) Which residual plots look like they're flat with constant width? For the ones which don't look like this, describe how they differ.

    Extra credit, 5 points: Add kernel smoothing lines to each of the residual plots. Comment.

2

. . . But We Make It Up in Volume

Source: Rodrik (2008)

"Gross domestic product" is a standard measure of the size of an economy; it's the total value of all goods and services bought and solid in a country over the course of a year. It's not a perfect measure of prosperity[1], but it is a very common one, and many important questions in economics turn on what leads GDP to grow faster or slower.

One common idea is that poorer economies, those with lower initial GDPs, should grower faster than richer ones. The reasoning behind this "catching up" is that poor economies can copy technologies and procedures from richer ones, but already-developed countries can only grow as technology advances. A second, separate idea is that countries can boost their growth rate by under-valuing their currency, making the goods and services they export cheaper.

This week's data set contains the following variables:

- Country, in a three-letter code (see `http://en.wikipedia.org/wiki/ISO_3166-1_alpha-3`).
- Year (in five-year increments).
- Per-capita GDP, in dollars per person per year ("real" or inflation-adjusted).
- Average percentage growth rate in GDP over the next five years.
- An index of currency under-valuation[2]. The index is 0 if the currency is neither over- nor under- valued, positive if under-valued, negative if it is over-valued.

Note that not all countries have data for all years. However, there are no missing values in the data table.

[1] A standard example: if vandals break all the windows on a street, a town, GDP goes *up* by the cost of the repairs.

[2] The idea is to compare the actual exchange rate with the US dollar to what's implied by the prices of internationally traded goods in that country — the exchange rate which would ensure "purchasing power parity". The details are in the paper this assignment is based on, which will be revealed in the solutions.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

1. (10) Linearly regress the growth rate on the under-valuation index and the log of GDP. Report the coefficients and their standard errors (to reasonable precision). Do the coefficients support the idea of "catching up"? Do they support the idea that under-valuing a currency boosts economic growth?
2. (20) Repeat the linear regression but add as covariates the country, and the year. Use `factor(year)`, not `year`, in the regression formula.
    1. (5) Report the coefficients for log GDP and undervaluation, and their standard errors, to reasonable precision.
    2. (5) Explain why it is more appropriate to use `factor(year)` in the formula than just `year`.
    3. (5) Plot the coefficients on year versus time.
    4. (5) Does this expanded model support the idea of catching up? Of under-valuation boosting growth?
3. (10) Does adding in year and country as covariates improve the predictive ability of a linear model which includes log GDP and under-valuation?
    1. (1) What are the R^2 and the adjusted R^2 of the two models?
    2. (5) Use leave-one-out cross-validation to find the mean squared errors of the two models. Which one actually predicts better, and by how much? *Hint:* Use the code from lecture 3.
    3. (4) Explain why using 5-fold cross-validation would be hard here. (You don't need to figure out how to do it.)
4. (20) *Kernel smoothing* Use kernel regression, as implemented in the `np` package, to non-parametrically regress growth on log GDP, under-valuation, country, and year (treating year as a categorical variable). *Hint:* read chapter four carefully. In particular, try setting `tol` to about 10^{-3} and `ftol` to about 10^{-4} in the `npreg` command, and allow several minutes for it to run. (If you are using R Markdown, trying caching this part of your code.)
    1. (5) Give the coefficients of the kernel regression, or explain why you can't.
    2. (5) Plot the predicted values of the kernel regression, for each country and year, against the predicted values of the linear model.
    3. (5) Plot the residuals of the kernel regression against its predicted values. Should these points be scattered around a flat line, if the model is right? Are they?
    4. (5) The `npreg` function reports a cross-validated estimate of the mean squared error for the model it fits. What is that? Does the kernel regression predict better or worse than the linear model with the same variables?
5. (20) *Time courses and interactions* In this question, use the kernel regression you fit in the previous problem.
    1. (6) Plot the predicted growth rate, as a function of the year, in five year increments from 1955 to 2000, if the initial GDP (not log GDP!) is \$10,000 in each period, the under-valuation index is 0 (i.e., no under- or over- valuation), and the country is Turkey.
    2. (3) Re-do the plot but change the under-valuation index to +0.5.

3. (3) Re-do the plot but hold the initial GDP at \$1,000 and the under-valuation index at 0.
4. (3) Re-do the plot with the initial GDP at \$1,000 and the under-valuation index at +0.5.
5. (5) Is there evidence of an interaction between initial GDP and under-valuation? Explain.

6. (20) *Average predictive comparisons* §[[4.5]] explains how to calculate the "average predictive comparison" — the typical rate of change in the response when a given variable is perturbed, even when the model is nonlinear and has interactions. See, in particular, Equation [[4.31]].

   Hint: at no point in this problem should you re-fit either model.

   1. (5) Calculate the average predictive comparison for log GDP in the kernel regression.
   2. (5) Calculate the average predictive comparison for under-valuation in the kernel regression.
   3. (5) Explain how to calculate the corresponding average predictive comparisons from the linear model's coefficients. What are the average predictive comparisons for initial log GDP and for under-valuation in the linear model?
   4. (5) Do the kernel and the linear regression agree, qualitatively, about the average effect of increasing initial GDP on growth? Do they agree, qualitatively, about the effect of undervaluation on growth?

3

Past Performance, Future Results

AGENDA: Practice with cross-validation and with smoothing; baby steps in using simulation to see how a model behaves and to do hypothesis testing; reinforcement that "the variable matters" $\neq$ "the coefficient on the variable is statistically significant".

TIMING: Some parts of this assignment, particularly Problem 66, are very computation-intensive. Start early, read the hints, and cache your results.

A corporation's **earnings** in a given year is its income minus its expenses[1]. The **return** on an investment over a year is the fractional change in its value, $(v_{t+1} - v_t)/v_t$, and the average rate of return over k years is $[(v_{t+k} - v_t)/v_t]^{1/k}$. Our data set this week looks at the relationship between US stock prices, the earnings of the corporations, and the returns on investment in stocks, with returns counting both changes in stock price and dividends paid to stock holders.[2]

Specifically, our data contains the following variables:

- Date, with fractions of a year indicating months
- Price of an index of US stocks (inflation-adjusted)
- Earnings per share (also inflation-adjusted);
- `Earnings_10MA_back`, a ten-year moving average of earnings, looking backwards from the current date;
- `Return_cumul`, cumulative return of investing in the stock index, from the beginning;
- `Return_10_fwd`, the average rate of return over the next 10 years from the current date.

"Returns" will refer to `Return_10_fwd` throughout.

[[TODO: link to data set]]

1. *Inventing a variable*
    1. (1) Add a new column, `MAPE`, to the data frame, which is the ratio of `Price` to `Earnings_10MA_back`. It should have the following summary statistics:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
  4.785  11.710  15.950  16.550  19.960  44.200     120
```

[1] Accountants get into subtle issues about whether to include in expenses taxes, interest paid on loans, and charges for depreciation of assets and amortization of investments. Those of you who get jobs with certain kinds of tech company will grow only too familiar with these wrinkles. In our data set, earnings are very definitely after all these expenses.

[2] Nothing in this assignment, or the solutions, should be taken as financial advice.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

Why are there exactly 120 NAs?

2. (1) Linearly regress the returns on `MAPE` (and nothing else). What is the coefficient and its standard error? Is it significant?
3. (1) What is the MSE of this model, under five-fold CV?

2. *Inverting a variable*

    1. (3) Linearly regress the returns on 1/`MAPE` (and nothing else). What is the coefficient and its standard error? Is it significant? (For full credit, do not add a new column to the data frame, or create a new vector.)
    2. (1) What is the five-fold CV MSE of this model? How does it compare to the previous one?

3. *Employing a variable* A simple-minded model[3] says that expected returns over the next ten years should be exactly equal to 1/`MAPE`.

    1. (1) Find the in-sample MSE of this model.
    2. (2) Explain why the in-sample MSE is an unbiased estimate of the generalization error for this particular model.
    3. (2) Make a $Q-Q$ plot for the residuals of this model. *Hint:* try subtraction, rather than `residuals`.
    4. (5) Estimate a t distribution from the residuals. Report the parameters and their standard errors. Plot a histogram of the residuals, and add the estimated t density. *Hint:* see the function `fitdistr` in the `MASS` package.

4. (5) Use `npreg` to estimate a kernel regression of the returns on `MAPE`. What is the bandwidth? The cross-validated MSE?
5. *One big happy plot* For this problem, you need to only include one plot, and one paragraph of writing, but make sure you clearly label, with comments, which parts of your code are answers to each question. (This does not mean showing your code in your report.) Also, in this problem, take "line" to mean "straight or curved line, as appropriate". Plotting disconnected points where a line is called for will get partial credit.

    1. (1) Make a scatter-plot of the returns against `MAPE`.
    2. (6) Add two lines, showing the predictions from the models you fit in problem 1 and 2.
    3. (1) Add a curve showing the predictions of the simple-minded model from problem 3.
    4. (5) Add a line of the predictions of the kernel regression to the plot from problem 5. Which of the previous models does it most resemble? Is it just a slightly wiggly copy of that model, or does it do something qualitatively different?

6. *Simulating the simple-minded model*

[3] Assume that: future earnings get added to the value of an investment in the company's stock; that nothing else adds to the value of the investment; and that earnings over the next ten years will be equal to those over the last ten years. Solve for the returns.

1. (10) Write a function which simulates the simple-minded model from problem 3. The function should take as inputs (i) a vector of `MAPE` values, and (ii) the three parameters of the t distribution. It should return a two-column data frame, with one column being `MAPE` and the other being 1/`MAPE` plus t-distributed noise. The columns should have names which match the names used in the real data frame. Make sure that the output of your function has the right number of rows and columns, and that the `summary` statistics for the two columns are what they should be (at least approximately, in the case of the second column).
2. (5) Write a function which takes as input a data frame, estimates the same linear model as in problem 2 to that data frame, and returns the coefficient on 1/`MAPE`. Check that it works by running it on the original data. Check that it also works when the input comes from your simulation function from 61.
3. (7) By repeated simulation, find the probability, under the simple-minded model, of the coefficient on 1/`MAPE` being as far from 1.0 (in either direction) as what you found in the data.
4. (8) You can now report a p-value for testing the hypothesis that this slope is exactly 1.0. Carefully state the null and alternative hypotheses, and give your p-value.
5. (7) Write a function which takes as input a data frame, estimates the same kernel regression as in problem 4, and returns the vector of fitted values from that regression. Check that it works by running it on your original data. Check that it also works when the input comes from your simulation function.
6. (8) Create a plot of predicted returns versus `MAPE` for the simple-minded model, as in problem 53. Add 200 kernel regression curves, fit to 200 simulations of the model. Finally, add the kernel regression curve from the true data, as in problem 54. (You'll want to manipulate graphics settings.) How plausible is the simple-minded model? Explain your answer by referring to your plot.
   Hint/warning: Estimating all the kernel regressions might well take a few seconds per simulation. Write and debug your code here with a smaller number of curves, then increase it for the final version.

7. *More fun with star-gazing*
   1. (1) Linearly regress the returns on both `MAPE` and 1/`MAPE` (without interaction). What are the coefficients? Which ones are significant?
   2. (1) Linearly regress the returns on `MAPE`, 1/`MAPE`, and the square of `MAPE`. What are the coefficients? Which ones are significant?
   3. (8) Explain what is going on.

4

Free Soil

AGENDA: Practice writing, testing, and debugging simple R functions. Practice decomposing a big computational problem into a bunch of small, inter-locking functions. Practice estimating a categorical contrast. Practice with weighted least squares. Practice with bootstrapping. Finally, an early observance of Lincoln's birthday.

Source: Chetty *et al.* (2014)

Recall that equation for the standard error of a proportion, when we observe a binomial with n trials and success probability p:

$$\sqrt{\frac{p(1-p)}{n}} \tag{4.1}$$

Further recall the estimated standard error in an observed proportion $\hat{p}$:

$$\sqrt{\frac{\hat{p}(1-\hat{p})}{n}} \tag{4.2}$$

Recall, finally, that the `Mobility` variable from homework 1 was an observed proportion, the fraction of children born into the bottom fifth of the income distribution who make their way to the top fifth of the distribution by age 30.

Load the data set from homework 1 as a data frame named `mobility`. We will only need three columns, `Mobility`, `Population` and `State`, though you may also want to keep `Name` for debugging purposes. Do not remove any row from the data frame which has complete values for these variables.

1. (15) Write a function, `se.prop`, to calculate the standard error for proportions. It should take a vector of proportions, `p`, and a vector of trial numbers, `n`, and return a vector of standard errors.
    1. (2) Construct a test case to check that `se.prop` gives the right answer when $p = 0.5$, $n = 1$.
    2. (2) Construct a test case to check that when `se.prop` is given a vector of different `n`'s, all with the same p (not equal to 0 or 1), the answers are proportional to $1/\sqrt{n}$.
    3. (2) Construct a test case to check that when $p = 0$, the returned value is always 0, for multiple n.
    4. (2) Construct a test case to check that when $p = 1$, the returned value is always 0, for multiple n.
    5. (2) Construct a test case to check that when given a vector `p` of mixed 0s and 1s, the returned vector has all 0s, for multiple `n`.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

6. (2) Construct a test case to check that when given a vector of different, non-extreme values for `p`, and a constant `n`, the entries of the returned vector are proportional to $\sqrt{p(1-p)}$.
7. (2) Check that `se.prop` works properly when `p=c(0.3,0.8)` and `n=c(12,72)`. This includes working out what the proper answers should be.
8. (1) Explain whether your code implements Eq. 1 or Eq. 2.

2. (10)
   1. (3) Use `se.prop` to calculate the standard error of the mobility for each community in the data from homework 1; report the summary statistics.
   2. (1) Plot the histogram of the standard errors.
   3. (2) Make a scatter-plot of the standard errors vs. population.
   4. (2) Make a scatter-plot of the standard errors vs. mobility.
   5. (2) How reliable were the inferential statistics you calculated in homework 1?

3. (15)
   1. (5) Write a function, `WSE`, to calculate weighted mean squared error. It should take as arguments `predicted`, a vector of predicted values; `observed`, a vector of observed values; and `weights`, a vector of weights. It should return a single real number, the weighted mean squared error. Mathematically, that is to say, it should find
   $$\frac{\sum_{i=1}^{n} w_i \left(y_i - \hat{y}_i\right)^2}{\sum_{i'=1}^{n} w_{i'}}$$
   Make the default value for `observed` the `Mobility` column of the data, and the default values for `weights` equal to one over the squares of the standard errors in `Mobility` from the previous problem. *Hint:* You could write this using a `for` loop, or even two of them, but there are more elegant ways.
   2. (3) Check that `WSE` works properly when `predicted` is `c(0.15,0.05)`, `observed` is `c(0.14,0.07)`, and `weights` is `c(0.01, 0.42)`. (This includes working out what the right answer should be.)
   3. (2) Create three modified versions of this test case, each changing one of the three arguments, and make sure that your function works correctly on all three.
   4. (2) Explain why, for modeling mobility, the weights should be the inverse square standard errors.
   5. (3) Check that `WSE` returns the MSE when all the weights are equal. (They will *not* be equal for those default values.)

4. (10)
   1. (5) Write a function, `dixie`, which reads in a vector of state names (in the form used in the mobility data set), and returns a binary vector, 1 if the state was part of the Confederacy during the US civil war, and 0 otherwise.
   2. (5) Check that it gives the correct results when applied to a vector of the 50 state names and the District of Columbia.

5. (10) Write a function, `dixie.fit`, which takes two arguments: a data frame with a column named `State`, and a vector of length two, `levels`. It should test, for each row, whether the state was in the Confederacy (using `dixie`), and if so return the first element of `levels`, and if not, return the second element. Check that it works correctly when `levels=c(1,0)`. Explain how you know that is the correct behavior.
6. (10) Write a function, `dixie.WSE`, which takes as input `levels`, without default, and a data frame, defaulting to `mobility`. It should predict the mobility level for each city based on whether it was in the Confederacy or not, using the function `dixie.fit`, and return the weighted squared error, using `WSE`, with the actual values of `Mobility` as the response and weights based on their standard errors. For full credit, call, do not re-write, the functions from the earlier problems.

   Construct a test case using a data frame of four rows to check that is working properly, when `levels=c(0.01,0.15)`.
7. (5) Optimize the weighted squared error for this two-parameter model, starting from the initial guess that the mobility level for the former Confederacy is 0.01, while that for the rest of the country is 0.15. Report the best-fitting values of `levels`.
8. (10) Turn the optimization from the previous problem into a function, which takes as arguments a data frame (with default equal to `mobility`) and an initial guess at `levels` (with default equal to `c(0.01,0.15)`), and returns the fitted values of `levels` (and nothing else). Check that running it with the defaults reproduces your answer from the previous problem. Check that you get a different answer if you remove the first half of the data frame.
9. (5) Use resampling of rows to give standard errors for `levels`.

Extra credit (10): Show, mathematically, that the optimal values for `levels` are always given by two weighted averages of `Mobility`. Show how to find them by two calls to `weighted.average`, without using `WSE`, `dixie.fit`, `dixie.WSE`, or any optimization function. For full extra credit, check that code implementing this matches the answer you obtained above.

5

There Were Giants in the Earth in Those Day

[[TODO: See if there're any improvements over previous versions; if not, cut]]

AGENDA: Explicitly: splines, bootstrap, simulation, comparing a simulation to data; implicitly: more practice writing, testing, and debugging simple functions.

Source: Clauset and Erwin (2008)

Some biologists argue that larger animals tend to have advantages over smaller members of their species, so that natural selection should tend to lead to an increase in size within an evolutionary lineage[1]. There is also some evidence that larger species tend to be shorter-lived than smaller ones[2]. In this assignment, we will look at the evidence for an increase in species size within lineages, and how the trade-off between these two forces might lead to a stable distribution of sizes across species.

We will use two data sets:

- The North American Mammalian Paleofauna Database (nampd.csv) lists, for about 2000 living and extinct species, the log of the mass, in grams, of a typical member of the species; the log mass of the ancestral species (when known); and the dates of the species' first and last appearance in the fossil record, in millions of years ago. If the last appearance date is `NA`, the species is still alive. This means you should *not* just throw away all rows containing NAs.
- The Masses of Mammals (MoM.txt) gives, for about 4000 living species, their mass in grams, identifying codes for the species, genus, and other taxonomic groups, and an indicator for whether the species lives in the land or in the water.

The model we will work with goes as follows: At any given time t, there is a collection of n_t species, whose masses are $X_1, X_2, \ldots X_{n_t}$. At each time step, one current species A gets picked, uniformly at random, to evolve into two new species. The masses of a descendant species X_D is related to that of its ancestor, X_A, by the model

$$X_D = \exp\left(r(\log X_A) + Z\right) \tag{5.1}$$

where $Z \sim \mathcal{N}(0, \sigma^2)$, and r is a function to be learned from the data, subject to

[1] Among other things, larger animals may be harder for predators to attack, find it easier to over-come prey or other members of their species, and be more efficient metabolically. For more, see, e.g., Bonner (1988).

[2] This may be because larger animals need more food in total, and possibly more specialized food sources, so they are more vulnerable to shifts in their environment.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

the restriction that X_D has to be at least $x_{\min}$ and at most $x_{\max}$. The ancestor X_A is removed from the current list of species, and its two independent descendants are added. After this, all species currently in the list have a risk of going extinct, with the probability for a species of mass x going extinct being a function of their mass,

$$p_e(x) = \beta x^{\rho} \tag{5.2}$$

Any species become extinct are removed from the collection. We then iterate the model again.

In all of the following questions, unless otherwise specified, you may take $\sigma^2 = 0.63$ (what are the units?), $x_{\min} = 1.8$ grams, $x_{\max} = 10^{15}$ grams, $\rho = 0.025$, and $\beta = 1/5000$.

1. (5) Linearly regress the log of the new mass on the log of the ancestral mass. Plot this regression line, along with a scatter-plot of the data, *in units of grams*, not log-grams. Carefully explain the interpretation of both the slope and the intercept. A rote recitation of "a one unit change", etc., will not receive full credit; think about the model, the transformations, and what the transformed model says about the variables.
2. (10) Use a smoothing spline to do a nonparametric regression of log new mass on log ancestral mass. Create a plot showing the data points, the model from question 1, and the spline, making sure that the axes are in units of grams, not log-grams.
3. (20)
    1. (10) Using resampling of residuals, calculate 95% confidence bands for the spline curve, and add them to the plot.
    2. (10) Using resampling of cases, calculate standard errors for the spline curve, and add bands at ± 2 standard errors to the plot.
4. (10) Write a function, `rmass`, which takes as inputs a single ancestral mass X_A (not $\log X_A$), an estimated spline function r, and any other parameters required by the model, and returns a single random value for X_D, according to Eq. 5.1. Make sure the returned value is in grams, not log grams. You will probably find it easiest to keep generating candidate values for X_D, until you get one which is between the limits. *Hint:* `while`
    1. (2) What model parameters does your `rmass` need?
    2. (4) Check, by repeated simulation, that the output is always between $x_{\min}$ and max, even when X_A is brought near either limit.
    3. (4) Using the spline curve you estimated in question 2, create 150 evenly spaced X_A values between $x_{\min}$ and max, generate an X_D for each of them, and fit a spline curve to the simulated values. Check that it is close to, but not identical with, the one you found from the data. (Why should it not be identical?)
5. (10) Write a function, `origin`, which takes the same arguments as `rmass`, except that instead of one ancestral mass it can take a vector of them. `origin` should

pick one entry from the vector to be X_A, and generate two independent values of X_D from it. One of these should replace the entry for X_A, and the other should be added to the end of the vector.

   1. (4) Check, by simulating with a length-one vector of ancestral masses, that neither component of the returned value matches the ancestral mass (why?), that both components have the same marginal distribution, and that the two components are uncorrelated with each other.
   2. (2) Check, by simulating, that if the input vector of masses has length m, the output vector always has length $m + 1$. (Check at least two values of m.)
   3. (4) Check, by simulating, that $m - 1$ entries in the output match the input exactly. Check this for at least two values of m. *Hint:* `is.element`, or `%in%`, or `match`.

6. (5) Write a function, `extinct.prob`, which takes as inputs a vector of species masses, and parameters ρ and β, and returns the extinction probabilities according to Eq. 5.2.

   1. (2) Check that if the masses are `c(100, 1600, 10000)` grams, $\rho = 1/2$ and $\beta = 1/200$, then `extinct.prob` returns the right values.
   2. (1) Check that if $\rho = 0$, the output probabilities are all β, no matter what the masses are.
   3. (1) Check that if the input masses are all equal, so are the returned probabilities, for at least three of different combinations of mass, ρ and β.
   4. (1) Check that if $\rho \neq 0$ and $\beta \neq 0$, and the masses are all different, then the returned probabilities are all distinct.

7. (5) Write a function, `extinction`, which takes a vector of species masses, ρ and β, and returns a possibly-shorter vector which removes the masses of species which were probabilistically selected for extinction. Be sure to handle the (unfortunate) case where every species goes extinct. *Hint:* Explain what `rbinom(n,size=1,prob=p)` does when `p` is a vector of length `n`.

   1. (1) Check that if $\beta = 0$, the output vector is always the same as the input vector.
   2. (3) Create a case where the input masses are all equal, and ρ and β are set so that the extinction probability should be 1/2. Check that the output is, on average, half as long as the input.
   3. (1) In the same test cases as the previous part, check that all the values in the new vector of masses were also in the old vector of masses.

8. (5) Write a function, `evolve_step`, which takes as inputs a vector of species masses, plus all needed parameters and estimated curves; calls `origin` and `extinction` as appropriate; and returns a new vector of species masses. How do you know it works?
9. (5) Write a function, `mass_evolve`, which takes the same inputs as `evolve_step`, plus an additional number `T`; iterates `evolve_step` T times; and returns the

final vector of species masses. How do you know it works? *Hint:* There will almost certainly need to be a `for` loop inside the function.

10. (5) In this question, use the default parameter values, and the spline you estimated in question 2.
    1. (1) Run `mass_evolve` starting from a single species with a mass of 120 grams for $T = 2 \times 10^5$ steps. Save the output as `masses.1`. Plot the histogram.
    2. (1) Re-run `mass_evolve` from the same conditions. Save as `masses.2`. Plot the histogram.
    3. (1) Re-run from the same conditions but for $T = 4 \times 10^5$ steps, saving as `masses.3`. Plot the histogram.
    4. (1) Change the starting condition to two species, one of 40 grams and one of 1000 grams. Run twice, both times with $T = 2 \times 10^5$, saving the results as `masses.4` and `masses.5`.
    5. (1) How do the distributions of the various `masses` compare to each other?
11. (5)
    1. (1) Load the Masses of Mammals data set, and plot the histogram of masses for land species.
    2. (2) Compare, verbally, the distribution for land species to that obtained from the simulations.
    3. (2) Compare the distributions using QQ plots.
12. (5) Does the output of the simulation model match the distribution of masses we actually observe? Are the differences between the model and reality bigger than those between different runs of the simulation? Are there qualitative distinctions between the simulation-to-simulation differences, and the simulation-to-reality differences? Support your answers by reference to the plots you have already made, or, if need be, new ones.

Note: more advanced techniques for comparing distributions exist (e.g., chapter F).

Extra credit: (10) Re-write the code so that Z, rather than being drawn from a Gaussian distribution, comes from resampling the residuals of the fitted spline curve. What do you have to modify? How much do the results change? Which version fits the observed mass distribution better?

6

The Sound of Gunfire, Off in the Distance

AGENDA: Explicitly, logistic models, generalized additive models, and checking regression specifications. Implicitly, the perils of science by p-value.

Sources: Collier and Hoeffler (2004) and Ward *et al.* (2010)

Our data this week, `http://www.stat.cmu.edu/~cshalizi/uADA/15/hw/06/ch.csv`, comes from a study of the causes of civil wars. Every row of the data represents a combination of a country and of a five year interval — the first row is `Afghanistan, 1960`, really meaning Afghanistan, 1960–1965. The variables are:

- The country name;
- The year;
- An indicator for whether a civil war *began* during that period — the code of `NA` means an on-going civil war, while 0 denotes continuing peace;
- Exports, really a measure of how dependent the country's economy is on *commodity* exports;
- Secondary school enrollment rate for males, as a percentage[1];
- Annual growth rate in GDP;
- An index of the geographic concentration of the country's population (which would be 1 if the entire population lives in one city, and 0 if it evenly spread across the territory);
- The number of months since the country's last war or the end of World War II, whichever is more recent[2];
- The natural logarithm of the country's population;
- An index of social "fractionalization", which tries to measure how much the country is divided along ethnic and/or religious lines;
- An index of ethnic dominance, which tries to measure how much one ethnic group runs affairs in the country.

Some of these variables are NA for some countries.

1. (10) Fit logistic regression for the start of civil war on all other variables except country and year; include a quadratic term for exports. Report the coefficients and their standard errors, together with R's p-values. Which ones does R say are significant at the 5% level?

[1] I have been unable to find an explanation anywhere of why this rate is greater than 100 for some data points.

[2] This appears to count only civil and not foreign wars.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

2. *Interpretation* (15) All parts of this question refer to the logistic regression model you just fit.
    1. (5) What is the model's predicted probability for a civil war in India in the period beginning 1975? What probability would it predict for a country just like India in 1975, except that its male secondary school enrollment rate was 30 points higher? What probability would it predict for a country just like India in 1975, except that the ratio of commodity exports to GDP was 0.1 higher?
    2. (5) What is the model's predicted probability for a civil war in Nigeria in the period beginning 1965? What probability would it predict for a country just like Nigeria in 1965, except that its male secondary school enrollment rate was 30 points higher? What probability would it predict for a country just like Nigeria in 1965, except that the ratio of commodity exports to GDP was 0.1 higher?
    3. (5) In parts (a) and (b), you changed the same predictor variables by the same amounts. If you did your calculations properly, the changes in predicted probabilities are not equal. Explain why not. (The reasons may or may not be the same for the two variables.)
3. *Confusion* (10) Logistic regression predicts a probability of civil war for each country and period. Suppose we want to make a definite prediction of civil war or not, that is, to **classify** each data point. The probability of mis-classification is minimized by predicting war if the probability is ≥ 0.5, and peace otherwise.
    1. (5) Build a 2×2 "confusion matrix" (a.k.a. "classification table" or "contigency table") which counts: the number of outbreaks of civil war correctly predicted by the logistic regression; the number of civil wars not predicted by the model; the number of false predictions of civil wars; and the number of correctly predicted absences of civil wars. (Note that some entries in the table may be zero.) Make sure the rows and columns of the table are clearly labeled.
    2. (3) What fraction of the logistic regression's predictions are correct? (Note that this is if anything too kind to the model, since it's an in-sample evaluation.)
    3. (2) Consider a foolish (?) pundit who always predicts "no war". What fraction of the pundit's predictions are correct on the whole data set? What fraction are correct on data points where the logistic regression model also makes a prediction?
4. *Calibration* (10) Divide the data points into groups where the predicted probability of a civil war is 0–10%, those where it is 10–20%, etc. Calculate the actual proportion of civil wars for each group of data points. Give a plot where the horizontal axis is the predicted probability, and the vertical is the actual frequency. Does the plot go up the 45-degree diagonal? Should it, if the model is right? If it does not, do observed frequencies at least increase as the predicted probability goes up, so that civil war really is more common when the

model says it has higher probability? (Again, this is if anything too kind to the logistic regression, because it's an in-sample comparison.)

5. (10) Fit a GAM with the same variables to the same data: smooth all the continuous predictor variables; do *not* include an explicit quadratic term for exports. (The ethnic-dominance variable is binary, and should be included in the model with `as.factor`.) Provide plots of the partial response functions. Which ones are at least roughly linear, and which are not?
6. (10) Calculate the confusion matrix for the GAM. What fraction of its predictions are accurate? How does that compare both to the logistic regression and the peace-always pundit?
7. (10) Repeat the calibration checking plot for the GAM. Are its probabilities closer to tracking actual frequencies, or further, than those of the logistic regression?
8. (15) Test whether the logistic regression is properly specified, using the GAM as the alternative model. (Follow the procedure in the notes.) What is the p-value? Explain, based on this test and any other results you have reported, which model you prefer.

Extra credit (15): Start with the model which predicts a constant probability of civil war for all countries and years. Evaluate its log-likelihood out of sample through five-fold cross-validation. Now consider all one-variable GAMs, using all available predictor variables except country and year. Which one variable has the highest cross-validation log-likelihood, and is it higher than the trivial, intercept-only model? Consider all two-variable GAMs which extend the one-variable GAM you just picked: report their cross-validated log-likelihoods. Are the two variables you picked the two variables with the smallest p-values in the logistic regression? Should they be?

7

The Bullet or the Ballot?

Many people assume that violence, while perhaps dangerous or evil, is more *effective* politically than non-violence. In this exam, we will examine whether, in fact, non-violent political movements are more or less likely to achieve their goals than violent ones. Moreover, we will look at the conditions which make non-violence more or less likely to succeed.

Our data set, gathered by political scientists who have studied exactly these questions, is `navc.csv` on the class website. The units of analysis here are political movements or campaigns. For each movement, the data records:

Source: Stephan and Chenowth (2008); Chenoweth and Stephan (2011)

- The name of the movement (`campaign`);
- The country the movement was in (`country`);
- The peak year of the movement's activity (`year`);
- Whether the movement fully achieved its aims (1.0), achieved partial success (0.5), or failed (0) (`outcome`);
- An indicator variable (`nonviol`), 1 for non-violent movements and 0 for others;
- A quantitative measure of how democratic the government of the country was, from -10 for very un-democratic governments to a possible maximum of +10 (`democracy`);
- An indicator for the government being under international sanctions (`sanctions`);
- An indicator for whether the government received aid from other governments to help deal with the movement (`aid`);
- An indicator for the movement's receiving aid from foreign governments (`support`);
- An indicator for the government's using violence to repress the movement (`viol.repress`);
- An indicator for whether substantial portions of the security (military and police) forces of the government sided with the movement (`defect`);
- The duration of the movement, in days (`duration`).

Specific analytic issues you must address

In general, are non-violent movements more likely to be successful than violent ones? Does violent repression by the government make movements more or less likely to be successful, and is there a difference in this effect between movements which are themselves violent and non-violent? Similarly, what is the effect of foreign aid to the government and to the movement? Do non-violent movements become more likely to succeed as the government becomes more democratic?

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

Does the difference in probability of success between violent and non-violent movements vary with how democratic the government is? All of these should be answered with reference to the results in your model (or models).

Models

Use a generalized additive model with a logistic link function; smooth all continuous predictor variables, and include all categorical variables, except campaign and country names, as your default. (Departures from this should be carefully justified.) Be sure to include the year as a predictor variable, and explain the interpretation of your estimated effects for the year. Some of the analytic issues above may be most easily addressed through including interaction terms, or through fitting different models on subsets of the data; describe any such variations, and the reasons for your choices.

Note 1: Before fitting a model with a logistic link function, you will need to re-code partial successes as either successes or failures. Explain which one you chose, and briefly justify your decision.

Note 2: The analysis could also be done with kernel models, and doing so would receive full credit, but computations may take too long. (This could however avoid needing to re-code partial successes.)

Inferential Statistics and Model Assessment

You may not assume that R's default standard errors or p-values on estimated regression coefficients can be trusted. Uncertainty should be assessed using suitable bootstrap or simulation procedures. (Be sure to explain why you used the procedure you did.) If you need to compare two models in terms of predictive accuracy, this should not be done through R's default significance tests or R^2's, but through either a suitable bootstrap or cross-validation (again, explain the reasoning behind your choices). Exceptions will be made if you can successfully argue that the default calculations are reliable *for this problem*.

Model checking

The answers you give to the substantive analytical questions rest on your estimated model, so you need to include some assessment of the model's goodness of fit. The exact way in which you do this is left up to your initiative; it may help to remember that the model is predicting probabilities of success. Be sure to describe your procedure and explain why you chose it, that is, why it is appropriate to answer the questions at hand.

Format

Your main report should be a humanly-readable document of at most 10 single-spaced pages, including figures. It should have the following sections:

INTRODUCTION describing the scientific problem and the data set, possibly including *relevant* summary statistics or exploratory graphs.

MODELS with subsections

- Describing the specification of the model (or models) you estimated, and explaining why you decided to use those specifications rather than others;
- Giving the relevant estimated coefficients and/or functions (possibly in visual form), along with suitable measures of uncertainty;
- Checking the goodness of fit of the model, including a description of the test procedures you used, why you chose those ways of checking the model, what the results were, and what they told you about the ability of the model to describe the data set.

RESULTS answering the analytical questions quantitatively, and with suitable measures of uncertainty, with reference to your estimated model or models.

You may assume that the reader has a general familiarity with the contents of 401, and with the models and methods we have covered so far in the course, but will need to be reminded of any details. The reader should not be assumed to have any prior familiarity with the data set.

Numerical results

Numerical quantities should be written out to appropriate precision, i.e., neither more nor fewer significant digits than appropriate.

Code

All statistical results must be supported by appropriate code, or they will receive no credit. ("Show your work.") The ideal would be to use R Markdown, or knitr+LaTeX, to embed all computations in a humanly readable document, and submit both the knitted version and the source[1] As a second best, it is acceptable to submit a PDF document containing all text and figures, and a separate .R file, containing all supporting computations, clearly labeled via the comments so that it is easy to see which claims or results go with which pieces of code.

Rubric

As usual, this describes the ideal.

Words

(10) The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

[1] See examples at `http://yihui.name/knitr/demos/`, and the useful chunk options like `echo` at `http://yihui.name/knitr/options/`.

Numbers

(5) All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.

Pictures

(5) Figures and tables are easy to read, with informative captions, axis labels and legends, and are placed near the relevant pieces of text.

Code

(15) The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the Markdown file knits (if applicable).

Modeling

(15) Regression model specifications are described clearly and in appropriate detail. There are clear explanations of how estimating the model helps to answer the analytical questions, and rationales for all modeling choices. If multiple models are compared, they are all clearly described, along with the rationale for considering multiple models, and the reasons for selecting one model over another, or for using multiple models simultaneously.

Inference

(20) The actual estimation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty.

Checking

(15) The goodness-of-fit of the model is actively probed by means of tests suitable to that class of model. The tests chosen are described, along with the rationale for using those tests. The execution of the tests is technically correct, and the results of the checks are clearly described. The extent to which the results of the model assessment build or undermine confidence in the conclusions is laid out clearly, with reference to the results of specific tests.

Conclusions

(15) The substantive, analytical questions are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about the model, or derived quantities, to substantive conclusions is both clear and

convincing. Contingent answers ("if X, then Y, but if Z, then W") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.

Extra credit

(10) Up to ten points may be awarded for reports which are unusually well-written, where the code is unusually elegant, where the analytical methods are unusually insightful, or where the analysis goes beyond the required set of analytical questions.

8

A Diversified Portfolio

WARNING: Some questions require slow computations.

Source: Classic material for finance, but see especially Fama and French (1993), and, for the history, MacKenzie (2006)

Classical financial theory suggests that the log-returns of corporate stocks should be IID Gaussian random variables, but allows for the possibility that different stocks might be correlated with each other. In fact, theory suggests that the returns to any given stock should be the sum of two components: one which is specific to that firm, and one which is common to all firms. (More specifically, the common component is one which couldn't be eliminated even in a perfectly diversified portfolio.) This in turn implies that stock returns should match a one-factor model.

The data file portfolio.csv consists of the log returns for the stocks of 22 selected large US corporations, centered to have mean zero and scaled to have standard deviation 1. Each row is labeled by the relevant date.

1. (10)
    1. (5) Report the weights of the first principal component. Since this is a vector of length 22, it will be better to report this visually than as a table or list of numbers. Comment on any notable patterns.
    2. (5) Plot the projection on to the first principal component against date. Comment on any notable patterns.
2. (10) Fit a one-factor model.
    1. (5) Report the vector of factor loadings. (Again, this will be most easily reported visually.) Comment on any notable patterns, and compare it to the first principal component.
    2. (5) Plot the factor score against the date. Comment on any notable patterns, and compare to the projection on the first principal component.
3. (10) Use case bootstrapping to provide 90% confidence intervals for the factor loadings of the one-factor model. Report the results as a figure rather than a table.
4. (5) What is the p-value of a goodness of fit test for the hypothesis that one factor is adequate? Explain carefully just what hypothesis is being tested, and what is entailed by rejecting or retaining it.
5. (5) Download the function charles from the class website. Explain carefully what arguments the function takes, what the function does, and exactly what

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

its return value is. (An acceptable answer to this question could be a thoroughly-commented version of the function.)

6. (15) Write a function which finds the cross-validated log-likelihood of a factor model with a given number of factors. That is, it should take a data set and a number of factors as inputs, divide the data randomly into folds, calculate the log-likelihood on a test fold of a model fit on the other folds, and return the average log-likelihood across folds. You are encouraged to re-use existing code from the solutions and notes; `charles` may or may not be useful. Report the five-fold cross-validated log-likelihood of factor models with from 1 to 10 factors for this data. What is the favored number of factors?
7. (10) Using the `mvnormalmixEM` function from the `mixtools` package, fit a two-component Gaussian mixture model to the data.
    1. (5) Report the parameters of the two mixture components, and their relative weights. Avoid excessive precision.
    2. (5) Use `posterior` component of the object returned by `mvnormalmixEM` to classify each day as belonging to one mixture component or the other. Plot the mixture components over time, and comment on any patterns.
8. (15) Write a function, `loglike.mvnormalmix`, which takes in a data set and a model returned by `mvnormalmixEM`, and returns a log-likelihood. Check that it works by seeing that it gives the correct value of the log-likelihood when a two-component mixture is fit to the whole data. (*Hint:* read section 21.4.4 of the notes.)
9. (8) Write a function which calculates the log-likelihood of mixture models through cross-validation, as in problem 6. Report the five-fold cross-validated log-likelihood of mixture models with from two to four components for this data. What is the favored number of mixture components?

    Warning: five-fold CV for four mixture components on the full data might take several hours. Start early, and make sure you debug your code on small parts of the data rather than the whole thing.
10. (2) Can you decide whether factor models or mixture models fit this data better?

Company	Abbreviation
Altria (formerly Philip Morris)	MO
Amazon	AMZN
Apple	AAPL
Archer Daniels Midland	ADM
Automatic Data Processing	ADP
Bank of America	BAC
Corrections Corporation of America	CXW
Dow Chemicals	DOW
Equifax	EFX
ExxonMobil	XOM
Ford	F
Halliburton	HAL
General Electric	GE
Goldman Sachs	GS
Graham Holding Companies	GHC
Microsoft	MSFT
Proctor and Gamble	PG
Time Warner	TWX
United States Steel	X
Walmart	WMT
Yahoo!	YHOO
Yum! Brands	YUM

Table 8.1 *Abbreviations for the companies included in the data set.*

9

The Monkey's Paw

[[TODO: Need to handle continuous vs. discrete issue here — perhaps make additional problem of constructing a Poissonian factor model? Or just too complicated?]]

Scientific background: Nerve cells (or "neurons") communicate and process information by transmitting little electrical impulses to each other, called "spikes"[1]. Many neurons use "rate codes", where the number of spikes they produce in a short period of time encodes information either about some aspect of the world the organism is sensing, or about how the organism is acting or is going to act.

For example, when very fine electrodes are inserted into certain motor-control regions of the brains of monkeys, so that neuroscientists can record from individual neurons, some cells are found to encode the direction in which the monkey intends to move its hand. Specifically, a neuron has a preferred direction vector $\vec{b}$, and the when the monkey intends to move its hand with velocity $\vec{v}$, the average number of spikes over a short interval is $a + \vec{b} \cdot \vec{v}$, plus or minus some amount of noise. A neuron which behaves like this is said to show "directional tuning", and $\vec{b}$ is its "preferred direction".[2]

The data set `neur.csv` is based on an experiment during which the neuroscientists recorded simultaneously from 96 directionally-sensitive neurons in a monkey's motor region, each cell having a different preferred direction. That is, each neuron i will have its own $\vec{b}_i$ and its own intercept a_i. During each trial, the monkey was to move its hand in one of eight directions, spread evenly around a circle. Each row of the data frame represents 100ms, and so the entries in the data frame are the number of spikes produced by each of the 96 neurons spiked during each time interval.

In this exam, you will both fit a model which derives from this "directional tuning" idea, and consider alternative multivariate models.

[1] Because of how they look in a plot of voltage against time.

[2] For more on such models of neural coding, see, for example, §3.3 of P. Dayan and L. F. Abbott, *Theoretical Neuroscience* (MIT Press, 2001).

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

9.1 Specific Problems

1. Explain how this model for spiking is, or is related to, a factor model. Your explanation should indicate how a, $\vec{b}$ and $\vec{v}$ are related to the factor loadings and factor scores, and the number of factors.
2. Fit a factor model with the number of factors you determined is appropriate from problem 1. For each neuron, report its preferred direction. (Since there are a large number of neurons, it would probably be best to report this visually.)
3. Based on your fitted factor model, report an estimate of the intended direction $\vec{v}$ at each time point. (Again, this should probably be reported visually.) The experiment had distinct breaks between trials where the monkey stopped moving in one direction and started moving in another, random direction; can you work out, approximately, where these breaks occurred?
4. Suppose that instead of recording intended velocities in the usual (x, y) coordinates, we used coordinate axes which were rotated 30 degrees counterclockwise from the usual ones. Show that this would amount to multiplying the intended-velocity vector $\vec{v}$ by $\begin{bmatrix} \cos \pi/6 & -\sin \pi/6 \\ \sin \pi/6 & \cos \pi/6 \end{bmatrix}$. Explain what effect, if any, this would have on the preferred-direction vector $\vec{b}$ of each neuron. Explain how this difference in coordinate systems could, or could not, be detected in your factor analysis of the data. In particular, what would this change of coordinates imply for the interpretation of your factor score estimates and factor loadings?
5. Try fitting a three-cluster mixture model. Why might three clusters, specifically, be reasonable? Which model predicts better, the factor model or the three-cluster mixture model?

   Note: if using the `mixtools` package, you might find it easier to use the function `npEM` to fit a non-parametric mixture model than to use `mvnormalmixEM` to fit a Gaussian mixture model, since the observable variables are discrete counts rather than continuous. Fitting such a mixture model to the full data may take as much as a couple of minutes, so allow plenty of time for debugging and any computation-intensive procedues.
6. Try fitting an eight-cluster mixture model. Why might eight clusters be reasonable? Which model predicts best? (See previous note.)

You are welcome to consider other models for this data as well, but for full credit you must answer all these questions about these models.

9.2 Formatting Instructions and Rubric

Your main report should be a humanly-readable document of at most 10 single-spaced pages, including figures. It should have the following sections:

INTRODUCTION describing the scientific problem and the data set, possibly including *relevant* summary statistics or exploratory graphs. (Do not include EDA just to have EDA.)

Specific Problems answering the questions set above, but avoiding the check-list, itemized format in favor of continuous text, with a logical succession of sentences and paragraphs. (Writing coherently is more important than following the order of the questions.)

Conclusions summarizing what you have learned from the data and models about whether the directional-tuning model is really a good description of how these neurons encode motion.

You may assume that the reader has a general familiarity with the contents of 401, and with the models and methods we have covered so far in the course, but will need to be reminded of any details. The reader should not be assumed to have any prior familiarity with the data set.

Numerical results

Numerical quantities should be written out to appropriate precision, i.e., neither more nor fewer significant digits than appropriate.

Code

All statistical results must be supported by appropriate code, or they will receive no credit. ("Show your work.") The ideal would be to use R Markdown, or knitr+LaTeX, to embed all computations in a humanly readable document, and submit both the knitted version and the source[3] As a second best, it is acceptable to submit a PDF document containing all text and figures, and a separate .R file, containing all supporting computations, clearly labeled via the comments so that it is easy to see which claims or results go with which pieces of code.

Rubric

As usual, this describes the ideal.

Words

(10) The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

Numbers

(5) All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.

[3] See examples at `http://yihui.name/knitr/demos/`, and the useful chunk options like `echo` at `http://yihui.name/knitr/options/`; also the examples in the solutions to exam 1.

Pictures

(5) Figures and tables are easy to read, with informative captions, axis labels and legends, and are placed near the relevant pieces of text.

Code

(15) The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the Markdown file knits (if applicable). The main text of the report is free of intrusive blocks of code, which are used only when a specifically-computational point is being made, or when code is actually the clearest way of describing a point.

Specific Problems

(25) All specific problems posed in §9.1 receive clear, well-written and correct answers. The answers show, and convey, a real grasp of the mathematical basis of the models being manipulated, and how quantities in the model are related to the underlying scientific questions about neural coding of movement.

Inference and Uncertainty

(15) The actual estimation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty (such as confidence intervals or standard errors).

Comparisons

(15) All comparisons between models are done in a statistically valid way: if in-sample, they are accompanied by an explanation of why this particular in-sample comparison will not lead to over-fitting; if out-of-sample, there is a clear description of the generalization process being performed. The execution of comparisons is technically correct, and their results clearly described. The extent to which comparisons provide either clear or ambiguous evidence about which models fit better is made plain to the reader, and is carried through to the ultimate conclusions.

Conclusions

(15) The substantive questions about neural coding are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about models, or derived quantities, to substantive conclusions is both clear and

convincing. Contingent answers ("if X, then Y, but if Z, then W") are likewise described as warranted by the model and data. If uncertainties in the data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.

Extra credit

(10) Up to ten points may be awarded for reports which are unusually well-written, where the code is unusually elegant, where the analytical methods are unusually insightful, or where the analysis goes beyond the required set of analytical questions.

10

What's That Got to Do with the Price of Condos in California?

AGENDA: As a warm-up and refresher in using linear regression to explore relationships between variables, we will look at a large data set on real estate prices.

The Census Bureau divides the country up into geographic regions, smaller than counties, called "tracts" of a few thousand people each, and reports much of its data at the level of tracts. This data set, drawn from the 2011 American Community Survey, contains information on the housing stock and economic circumstances of every tract in California and Pennsylvania. For each tract, the data file records a large number of variables (not all of which will be used in this assignment):

- A geographic ID code, a code for the state, a code for the county, and a code for the tract
- The population, latitude and longitude of the tract
- Its name
- The median value of the housing units in the tract
- The total number of units and the number of vacant units
- The median number of rooms per unit
- The mean number of people per household which owns its home, the mean number of people per renting household
- The median and mean income of households (in dollars, from all sources)
- The percentage of housing units built in 2005 or later; built in 2000–2004; built in the 1990s; in the 1980s; in the 1970s; in the 1960s; in the 1950s; in the 1940s; and in 1939 or earlier
- The percentage of housing units with 0 bedrooms; with 1 bedroom; with 2; with 3; with 4; with 5 or more bedrooms
- The percentage of households which own their home, and the percentage which rent

Remember that these are not values for individual houses or families, but summaries of all of the houses and families in the tract.

The basic question here has to do with how the quality of the housing stock, the income of the people, and the geography of the tract relate to house values in the tract. We will look at several different linear models, and see if they have reasonable interpretations, and/or make systematic errors.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

1. (3 pts) Not all variables are available for all tracts. Remove the rows containing NA values. All subsequent problems will be done on this cleaned data set. *Hint:* Recipe 5.27.
    1. (1) How many tracts are eliminated?
    2. (1) How many people live in those tracts?
    3. (1) What happens to the summary statistics for median house value and median income?
2. (7) *House value and income*
    1. (1) Linearly regress median house value on median household income. Report the intercept and the coefficient (to reasonable precision), and explain what they mean.
    2. (2) Regress median house value on mean household income. Report the intercept and the coefficient (to reasonable precision), and explain what they mean. Why are the coefficients for two different measure of household income different?
    3. (4) Regress median house value on both mean and median household income. Report the estimates, and interpret the coefficients, as before. Does this interpretation seem reasonable? Explain.
3. (10) Regress median house value on median income, mean income, population, number of housing units, number of vacant units, percentage of owners, median number of rooms, mean household size of homeowners, and mean household size of renters. Report all the estimated coefficients and their standard errors to reasonable precision, and explain what they mean. Why are the coefficients on income different from in the previous models?
4. (5) Which three variables are most important, in this model, for predicting house values? Explain your reasoning for deciding on this. *Hint:* make sure your answers wouldn't change if we changed the units of measurement for the predictor variables.
5. (20) *Checking residuals* for the model from problem 3.
    1. (5) Make a $Q - Q$ plot of the regression residuals.
    2. (5) Make scatter-plots of the regression residuals against each of the predictor variables, and add kernel smoother curves (as in Chapter 1). Describe any patterns you see. (A *very* rough rule of thumb is that the bandwidth should be about $\sigma n^{-1/5}$, where σ is the standard deviation of the predictor variable and n is the sample size.)
    3. (5) Make scatter-plots of the squared residuals against each of the predictor variables, and add kernel smoother curves. Describe any patterns you see.
    4. (5) Explain, using these plots, whether the residuals appear Gaussian and independent of the predictors.
6. (12) Fit the model from 3 to data from California alone, and again to data from Pennsylvania alone.
    1. (5) Report the two sets of coefficients and standard errors. Explain whether or not it is plausible that the true coefficients are really equally.

2. (2) What are the square root of the mean squared error (RMSEs) of the Pennsylvania and California coefficients, on their own data?
3. (5) Use the California coefficients to predict the Pennsylvania data. What is the RMSE? What is the correlation between the California coefficients' predictions for Pennsylvania, and the Pennsylvania coefficients' predictions? *Hint:* Recipe 11.18.

7. (10) Make a map of median house values. The vertical coordinate should be latitude, the horizontal coordinate should be longitude, and the house value should be indicated either by the color of the points (*Hint:* recipe 10.23), or by using a third dimension in a perspective plot. Describe the patterns that you see.
8. (10) Make a map of the regression residuals for the model from problem 3. Are they randomly scattered over space, or are there regions where the model systematically over- or under- predicts? Are there regions where the errors are unusually large in both directions? (You might also want to make a map of the absolute value of the residuals.) — If you cannot make a map, you can still get partial credit for scatter-plots of residuals against latitude and longitude.
9. (8) Fit a linear regression with all the variables from problem 3, as well as latitude and longitude. Report the new coefficients and their standard errors. What do the coefficients on latitude and longitude mean? How important are latitude and longitude in this new model?
10. (5) Make a map of the regression residuals for the new model from problem 9. Compare and contrast it with the map of the residuals from the previous model. Are the new residuals spatially uniform, or are there patterns?
11. (10) Regress the *log* of median house value on the same variables as in problem 9. Which model more accurately predicts housing prices? How can you tell?

11

The Advantages of Backwardness

Many theories of economic growth say that it's easier for poor countries to grow faster than rich countries — "catching up", or the "advantages of backwardness". One argument for this is that poor countries can grow by copying existing, successful technologies and ways of doing business from rich ones. But rich countries are already using those technologies, so they can only grow by finding new ones, and copying is faster than innovation. So, all else being equal, poor countries should grow faster than rich ones. One way to check this is to look at how growth rates are related to other economic variables.

Our data for examining this will be taken from the "Penn World Table" (`http://pwt.econ.upenn.edu/php_site/pwt_index.php`), for selected countries and years. The data file is `penn-select.csv` on the class website. Each row of this table gives, for a given country and a five-year period, the starting year, the initial population of the country, the initial gross domestic product (GDP)[1] per capita (adjusted for inflation and local purchasing power), the average annual growth rate of GDP over that period, the average population growth rate, the average percentage of GDP devoted to investment, and the average percentage ratio of trade (imports plus exports) to GDP[2].

We will use the `np` package on CRAN to do kernel regression.[3] Install it, and load the data file `penn-select.csv` (link on the class website).

1. (5 points) Fit a linear model of `gdp.growth` on `log(gdp)`. What is the coefficient? What does it suggest about catching-up?
2. (5 points) Fit a linear model of `gdp.growth` on `log(gdp)`, `pop.growth`, `invest` and `trade`. What is the coefficient on `log(gdp)`? What does it suggest about catching-up?
3. (5 points) It is sometimes suggested that the catching-up effect only works for countries which are open to trade with, and learning from, more-developed economies. Add an interaction between `log(gdp)` and `trade` to the model

[1] Annual gross domestic product is the total value of all goods and services produced in the country in a given year. It has some pathologies — an earthquake which breaks everyone's windows could *increase* GDP by the value of the repairs — but it's a standard measure of economic output.

[2] The Penn tables call this variable "openness". It can be bigger than 100, if, for instance, a country re-exports lots of its imports.

[3] In addition to the examples in Chapter 4, the package has good help files, and a tutorial at `http://www.jstatsoft.org/v27/i05`.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

from Problem 2. What are the relevant coefficients? What do they suggest about catching-up?

4. (15 points) Use data-set splitting, as in Chapter 3 of the notes, to decide which of these three linear models predicts best. (You can adapt the code from that chapter or write your own.) Which one is the winner?
5. (15 points) The `npreg` function in the `np` package does kernel regression. By default, it uses a combination of cross-validation and sophisticated but very slow optimization to pick the best bandwidth. In this problem, we will force it to use fixed bandwidths, and do the cross-validation ourselves.

   ```
   penn.0.1 <- npreg(gdp.growth~log(gdp),bws=0.1,data=penn)
   ```

   does a kernel regression of `growth` on `log(gdp)`, using the default kernel (which is Gaussian) and bandwidth 0.1. (You don't have to call the data `penn`.) You can run `fitted`, `predict`, etc., on the output of `npreg` just as you can on the output of `lm`. (There are more examples of using `npreg` in Chapter 4.)

   The code at the end of this assignment (also online) uses five-fold cross-validation to estimate the mean-squared error for the six bandwidths $0.05, 0.1, 0.2, 0.3, 0.4, 0.5$. Use it to create a plot of cross-validated MSE versus bandwidth. Add to the same plot the in-sample MSEs of those six bandwidths on the *whole* data. What bandwidth predicts best?
6. (10 points) Make a scatterplot of `log(gdp)` versus `growth`. Add the line for the linear model from problem 1. Add the fitted values for the kernel curve with the best bandwidth (according to the previous problem). What does this suggest about catching up?

   (There are at least two ways to get the fitted values for the kernel regression, using `fitted` or `predict`.)
7. (5 points) `npreg` will also do kernel regressions with multiple input variables. This time, use the built-in bandwidth selector:

   ```
   penn.npr <- npreg(gdp.growth ~ log(gdp) + pop.growth + invest
     + trade, data=penn, tol=0.1, ftol=0.1)
   ```

   (The last two arguments tell the bandwidth selector not to try very hard to optimize; it may still take several minutes.) What are the selected bandwidths? (Use `summary`.)
8. (5 points) Explain why we cannot add an interaction between `log(gdp)` and `trade` to the nonparametric regression from the previous problem.
9. (15 points) Sub-divide the data into points where the initial GDP per capita is $\leq$ \$700 and those where it is above. For each data point, use the kernel regression from problem 7 to predict the *change* in growth-rate from a 10% decrease in initial GDP (not a 10% decrease in log-GDP). Report the averages over the initially-poorer and the initially-richer data points. Describe what this suggests about catching up.

   Hints: use `predict()` with partially-modified data; *do not* estimate another

regression with artificially-lowered initial GDPs; make sure you are changing initial GDP by 10%, and not changing the log of GDP by 10%.

10. (10 points) To chose between the best linear model (as picked by you in problem 4) and the kernel regression from problem 7, use cross-validation again. Modify the code provided to use five-fold cross-validation to get CV MSEs for both the linear regression and for the nonparametric regression (with automatic bandwidth selection). Which predicts better?
11. (10 points) Based on your analysis, does the data support the idea of catching up, undermine it, support its happening under certain conditions, or provide no evidence either way? (As always, explain your answers.)

```
# Compare predictive ability of different bandwidths using k-fold CV
  # Inputs: number of folds, vector of bandwidths, dataframe
  # Presumes: data frame contains variables called "gdp.growth" and "gdp"
  # Output: vector of cross-validated MSEs for the different bandwidths
  # The default bandwidths here are NOT good ones for other problems
cv.growth.folds <- function(nfolds=5, bandwidths=c(0.05,(1:5)/10), df=penn) {
  require(np)
  case.folds <- rep(1:nfolds,length.out=nrow(df))
    # divide the cases as evenly as possible
  case.folds <- sample(case.folds) # randomly permute the order
  fold.mses <- matrix(0,nrow=nfolds,ncol=length(bandwidths))
  colnames(fold.mses) = as.character(bandwidths)
     # By naming the columns, we'll won't have to keep track of which bandwidth
     # is in which position
  for (fold in 1:nfolds) {
    # What are the training cases and what are the test cases?
    train <- df[case.folds!=fold,]
    test <- df[case.folds==fold,]
    for (bw in bandwidths) {
      # Fit to the training set
      # First create a "bandwidth object" with the fixed bandwidth
      current.npr.bw <- npregbw(gdp.growth ~ log(gdp), data=train, bws=bw,
        bandwidth.compute=FALSE)
      # Now actually use it to create the kernel regression
      current.npr <- npreg(bws=current.npr.bw)
      # Predict on the test set
      predictions <- predict(current.npr, newdata=test)
      # What's the mean-squared error?
      fold.mses[fold,paste(bw)] <- mean((test$gdp.growth - predictions)^2)
      # Using paste() here lets us access the column with the right name...
    }
  }
  # Average the MSEs
  bandwidths.cv.mses <- colMeans(fold.mses)
  return(bandwidths.cv.mses)
}
```

12

It's Not the Heat that Gets You, It's the Sustained Conjunction of Heat with Elevated Levels of Atmospheric Pollutants

AGENDA: More practice with additive models; more practice with transformed variables; extending additive models to include interactions; re-shaping data frames; answering "what if?" questions using models.

TIMING: Problems 1–4 and 6 involve fitting models to data, plotting, and interpretation, but no coding. Problem 5 requires explaining and using some provided code. Problem 7 requires doing some math, and possibly writing some code to do the corresponding calculation. The solutions for problems 1–7 take a few minutes to knit. The extra credit takes about 40 minutes to run with streamlined code.

The data set `chicago`, in the package `gamair`, contains data on the relationship between air pollution and the death rate in Chicago from 1 January 1987 to 31 December 2000. The seven variables are: the total number of (non-accidental) deaths each day (`death`); the median density over the city of large pollutant particles (`pm10median`); the median density of smaller pollutant particles (`pm25median`); the median concentration of ozone (O_3) in the air (`o3median`); the median concentration of sulfur dioxide (SO_2) in the air (`so2median`); the time in days (`time`); and the daily mean temperature (`tmpd`).

We will model how the death rate changes with pollution and temperature. Epidemiologists tell us that risk factors usually multiply together rather than adding, so we will fit additive models to the logarithm of the number of deaths. For fitting additive models, please use the `mgcv` package.

1. Load the data set and run `summary` on it.
    1. (1) Is temperature given in degrees Fahrenheit or degrees Celsius?
    2. (2) The pollution variables are negative at least half the time. What might this mean?
    3. (2) We will ignore the `pm25median` variable in the rest of this problem set. Why is this reasonable?
2. Fit a spline smoothing of `log(death)` on time. (You can use either `smooth.spline` or `gam`.)
    1. (4) Plot the smoothing spline along with the actual values.
    2. (3) There should be four large outliers, right next to each other in time. When are they? For full credit, give calendar dates, not day numbers. (*Hints:* day 0 was 31 December 1993; the `as.Date` function.)
    3. (3) Calculate the R^2 of the model. In what sense, if any, is this the "proportion of variance *explained* by the model"?

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

3. Use `gam` to fit an additive model for `log(death)` on `pm10median`, `o3median`, `so2median`, `tmpd` and `time`. Use spline smoothing for each of these predictor variables. *Hint:* Because of some missing-data issues, some plots later may be easier to make if you set the `na.action=na.exclude` option when estimating the model.
    1. (7) Plot the partial response functions, with partial residuals. Describe the partial response functions in words.
    2. (4) Plot the fitted values as a function of time, along with the actual values of `log(death)`. *Hint:* You will have to be careful about the NA values!
    3. (4) Are the outliers still there? Are they any better? *Hint:* Look at the residuals here.
4. Medically, it makes more sense to suppose that deaths on day t are due conditions over the previous few days, and not just on the conditions on day t. This problem re-shapes the data set to let us model this.
    1. (8) Suppose that on any given day, we want to know the average value of some variable over today and the previous k days. Explain how the following code computes that.

```
lag.mean <- function(x, window) {
  n <- length(x)
  y <- rep(0,n-window)
  for (t in 0:window) {
    y <- y + x[(t+1):(n-window+t)]
  }
  return(y/(window+1))
}
```

    In particular, how is k related to the arguments?
    2. (7) Create a new data frame with the same column names as `chicago`, but where, on each day, the value of the pollution concentrations and temperature is the average of that day's value with the previous three days. (*Hint:* you will want to do different things to different columns of `chicago`.) How many rows should this data frame have? Make sure that the `time` and `death` columns are properly aligned with the new, time-average predictor variables. How can you check that this is working properly?
5. Fit an additive model, as in problem 3, with the time-averaged pollution and temperature variables. (Do not average `time` or `death`.)
    1. (5) Plot the partial response functions and their partial residuals.
    2. (5) Plot the fitted values as a function of time, and the actual values. What has happened to the outliers? *Hint:* Again, look at the residuals.
6. *Variable examination*
    1. (4) Find the rows in the data frame (with the time-averaged values) corresponding to the large-death outliers. Look at all variables for them, and for three days on either side. Now compare this to the same stretch of time a

year earlier. Which two variables, aside from `death`, are unusually high or low around the outliers?
2. (7) Re-fit the model from problem 5, with an interaction between the two variables you just picked out. Plot the partial response functions.
3. (4) Plot the fitted values versus time. What has happened to the outliers? *Hint:* Residuals once more.

7. Using the last model you fit, we will consider the predicted impact of a 2° Celsius increase in temperature on `log(death)`, taking the last full year of the data as a baseline.[1].
   1. (1) Prepare a data frame containing only the last full year of the data. What is the average predicted value of `log(deaths)`?
   2. (1) Modify this data frame to increase all temperatures by 2°C.
   3. (3) Find the new average *change* in the predicted values of `log(deaths)` associated with a 2°C warming.
   4. (5) Find a standard error for this average predicted change, using the standard errors for the prediction on each day, and assuming no correlation among them. Include an explanation of why your calculation is correct. Also give the corresponding Gaussian 95% confidence interval. *Hint 1:* The `se.fit` option to `predict`. *Hint 2:* The appendix to the textbook on "propagation of error".
   5. (5) Find the predicted change in the number of deaths (not change in `log(death)` from a 2°C warming over the course of a whole year. *Hint:* remember that $\overline{e^x} \neq e^{\overline{x}}$.
   6. (5) Find a standard error for the predicted change in the number of deaths (not the change in `log(death)`) and the corresponding 95% Gaussian confidence interval. *Hint:* Propagation of error again.

Extra credit 1 (10):

1. (4) Explain how you could use bootstrapping to give a 95% confidence interval for the average increase in `log(death)` over the year. Explain how your idea will handle the fact that the model uses multiple variables, and that what happens on day t is not independent of what happens on day $t-1$. More credit will be given for more precise, complete and clear explanations. (You do not have to implement your solution yet.)
2. (6) Implement your bootstrapping scheme and give the confidence interval.

[1] 2°C is in the middle range of current projections for the global average effect of climate change by the end of this century (`http://www.ipcc.ch/publications_and_data/ar4/wg1/en/contents.html`)q. Of course it's unrealistic to suppose that would be an even shift throughout the year, or for that matter that Chicago would necessarily warm by the average amount. In fact, some of the models (`http://www.ipcc.ch/publications_and_data/ar4/wg1/en/ch11s11-5-3.html`, Figure 11.11) have 4°C of warming in the middle of their prediction intervals for central North America.

13

Nice Demo City, but Will It Scale?

[[TODO: Integrate the two versions of this problem set, perhaps by just picking one]]

This version was used as a take-home exam, hence less scaffolding; flag as such in the guide to the problems

[[TODO: Yank references from *Preface to Urban Economics*]]

13.0.1 Version 1

13.0.1.1 Background

It has been known for a long time that larger cities tend to be more economically productive than smaller ones. That is, the economic output per person of a city or other settlement (Y) tends to increase with the population (N). Recently, there has been some controversy over the exact form of the relationship, and over its explanation.

In particular, it has been claimed[1] that urban incomes show "power-law scaling", meaning that

$$Y \approx y_0 N^a$$

for some constant $y_0 > 0$ (the same across cities) and some *scaling exponent* $a > 0$ (the same across cities). Equivalently[2],

$$\log Y \approx c + a \log N$$

The scientists who first postulated power law scaling for urban economies thought that the tendency for bigger cities to be more productive was largely due to what are called "increasing returns to scale"[3], which would be stronger in larger cities. Additionally, having more people around, and more different sorts of people around, could lead to exchanges of ideas and so to new and better ways of doing business. According to this view, the primary determinant of a city's economy is simply its size, and larger cities are just "scaled up" versions of smaller ones.

An alternative explanation is that different industries have different levels of income per worker, and that some industries tend to be concentrated in larger cities and others in smaller towns. Large cities tend especially to be the places where one finds highly skilled providers of very specialized services, though their services are used, often indirectly, more or less everywhere[4]. In this view, the

[1] By Geoffrey West and collaborators; there's a good video online of Prof. West giving a talk about the work at a TED conference, if you're interested.

[2] Why is it equivalent, and how is c related to y_0?

[3] This is when the cost of producing the same item, with the same factory, employees, etc., is lower when the volume being produced is high, perhaps because the system runs more efficiently, or each sale has to recover a smaller share of the fixed cost of setting up the factory. A constant sale price minus lower costs equals higher profits.

[4] There are probably few, if any, electrochemical engineers who design liquid crystal displays working

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

association between the population of cities and their economic productivity is due to the kind of industries that go with being big cities, not some effect of size as such. There is no reason, according to this "urban hierarchy" view, why the relationship between per-capita income Y and urban population N should be a power law. In fact, the urban-hierarchy model doesn't even specify a particular functional relationship between how much of a city's economy comes from high-value industries and the city's income, just that the relationship is increasing.

Note that neither the power-law nor the urban-hierarchy model predicts Gaussian distributions.

In this exam, you will assess the evidence for power law scaling, and whether the "urban hierarchy" idea can explain the relationship between income and population.

Data

For data-collection purposes, urban regions of the United States are divided into several hundred "Metropolitan Statistical Areas" based on patterns of residence and commuting; these cut across the boundaries of legal cities and even states. In the last decade, the U.S. Bureau of Economic Analysis has begun to estimate "gross metropolitan products" for these areas — the equivalent of gross national product, but for each metropolitan area. (See Homework 2 for the definition of "gross national product".) Our data set contains the following variables, derived from the BEA:

- the name of each metropolitan area;
- its per-capita gross metropolitan product, in dollars (Y);
- its population (N);
- the share of its economy derived from finance (as a fraction between 0 and 1);
- the share of "professional and technical services";
- the share of "information, communication and technology" (ICT);
- and the share of "management of firms and enterprises".

Note that the last four columns have some missing values (`NA`s), since the BEA does not release those figures when doing so would disclose sensitive information about individual companies.

13.0.1.2 Tasks and Questions

You are to write a report assessing the (1) whether the power-law scaling model accurately represents the relationship between urban population and urban per-capita income; (2) whether, as the "urban hierarchy" idea implies, the relationship can be explained away by controlling for which industries are found in which cities; and (3) whether the power-law scaling or the urban-hierarchy idea provides a better model of urban economies.

Your report should have the following sections: an introduction, laying out

in Altoona, PA, but everyone there who buys a cellphone indirectly pays for the time and training of such engineers who live elsewhere.

the questions being investigated and the approach taken; a description of the data; detailed analyses; and conclusions. Your report should deal with *at least* the following specific points:

- The estimation of the scaling exponent a from the data, including its uncertainty[5];
- An estimate of the out-of-sample error of the power-law-scaling model;
- An examination of that model's residuals;
- A comparison of that model to non-parametric models of the size-income relationship (including, but not limited to, out-of-sample errors);
- Whether larger cities tend to have higher shares of the four high-value industries measured in the data set, and if so, what the size-industry relationship is;
- Whether cities with higher shares for those industries have higher incomes, and if so, what the industry-income relationship is;
- Whether, and in what sense, the income-industry relationships can explain the size-income relationship;
- How missing values were handled, and why;
- Appropriate quantifications of uncertainty for all estimates and hypothesis tests.

Adequately dealing with these points may, of course, lead to others.

13.0.2 Version 2

13.0.2.1

[[This version was a pair of homework assignments, so the points add up to 200]]

For data-collection purposes, urban areas of the United States are divided into several hundred "Metropolitan Statistical Areas" based on patterns of residence and commuting; these cut across the boundaries of legal cities and even states. In the last decade, the U.S. Bureau of Economic Analysis has begun to estimate "gross metropolitan products" for these areas — the equivalent of gross national product, but for each metropolitan area. (See Homework 2 for the definition of "gross national product".) Even more recently, it has been claimed that these gross metropolitan products show a simple quantitative regularity, called "supralinear power-law scaling". If Y is the gross metropolitan product in dollars, and N is the number of people in the city, then, the claim goes,

$$Y \approx cN^b \tag{13.1}$$

where the exponent $b > 1$ and the scale factor $c > 0$. This homework will use the tools built so far to test this hypothesis.

1. (15 points) A metropolitan area's gross per capita product is $y = Y/N$. Show that if Eq. 13.1 holds, then

$$\log y \approx \beta_0 + \beta_1 \log N$$

[5] *Hint:* You should get a value in the range $(0, 0.5)$.

How are β_0 and β_1 related to c and b?

2. (15 points) The data files `gmp_2006.csv` and `pcgmp_2006.csv` on the class website contain the total gross metropolitan product (Y) in millions of dollars, and the per capita gross metropolitan product (y) in dollars, for all metropolitan areas in the US in 2006. Read them in and use them to calculate the metropolitan populations (N). If it's done correctly, then running `summary` on the population figures should give

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
  54980  135600  231500  680900  530900 18850000
```

   (Your exact results may differ very slightly because of rounding and display settings.) What is the variance of $\log y$?
3. (20 points) *Estimating the power-law scaling model.* Use `lm` to linearly regress log per capita product, $\log y$, on log population, $\log N$. How does estimating this statistical model relate to Equation 13.1? What are the estimated coefficients? Are they compatible with the idea of supra-linear scaling? What is the mean squared error for $\log y$?
4. (15 points) Plot per capita product y against N, along with the fitted power-law relationship from problem 3. (Be careful about logs!)
5. (15 points) Fit a non-parametric smoother to $\log y$ and $\log N$. (You can use kernel regression, a spline, or any other non-parametric smoother.) What is the mean squared error for $\log y$? Describe, in words, how this curve compares to the power-law model from problem 3.
6. (20 points) Using the method from [[lecture 10, section 1]], test whether the power-law relationship is correctly specified. What is the p-value? What do you conclude about the validity of the power-law model, based not just on this problem but the previous ones as well?

13.0.2.2

We continue to investigate the relationship between how big cities are, and how economically productive they are. The scientists who first postulated power laws for urban economies thought that the tendency for bigger cities to be more productive was largely due to what are called "increasing returns to scale"[6], which would be bigger in larger cities. Additionally, having more people around, and more different sorts of people around, could lead to exchanges of ideas and so to new and better ways of doing business.

An alternative explanation is that different industries have different levels of income per worker, and that some industries tend to be concentrated in larger cities and others in smaller towns. Large cities tend especially to be the places where one finds highly skilled providers of very specialized services, though their

[6] This is when the cost of producing the same item, with the same factory, employees, etc., is lower when the volume being produced is high, perhaps because the system runs more efficiently, or each sale has to recover a smaller share of the fixed cost of setting up the factory. A constant sale price minus lower costs equals higher profits.

services are used, often indirectly, more or less everywhere[7]. In this view, the association between the population of cities and their economic productivity is due to the kind of industries that go with being big cities, not some effect of size as such.

In this exam, you will do a fairly simple test of these two explanations.

Data

A data file has been e-mailed to you at your Andrew account. It is a comma-separated text file (CSV), containing the following columns, in order, for each metropolitan area:

- the name of the metropolitan area;
- its per-capita gross metropolitan product (in dollars)
- its population;
- the share of its economy derived from finance (as a fraction between 0 and 1);
- the share of "professional and technical services";
- the share of "information, communication and technology" (ICT);
- and the share of "management of firms and enterprises".

The first three columns you saw in the last homework. The last four columns came from the same source. However, those columns have some missing values (`NA`s), since the Bureau of Economic Analysis does not release the data when doing so would disclose sensitive information about individual companies.

13.0.2.3 Problems

1. *More specialist service industries in bigger cities?*
    1. (2 points) For each of the four industries, create a scatter-plot of the share of that industries in the economy as a function of population. If a city is missing a value for an industry, omit it from that plot.
    2. (5 points) Add a nonparametric smoothing curve to each plot. Use kernel regression, local linear regression, a smoothing spline, etc., as you wish, but make sure that you use cross-validation to adapt the amount of smoothing to the roughness of the data.
    3. (3 points) Describe the patterns made by these plots. In particular, do larger cities have more of these industries?
2. *Higher productivity from specialist service industries?*
    1. (2 points) For each of the four industries, create a scatter-plot of per-capita GMP as a function of the share of that industry in the city's economy. If a city is missing a value for an industry, omit it from the plot.
    2. (5 points) Add a nonparametric smoothing curve to each plot. (Use the same smoothing method you did for problem 1.)

[7] There are probably very few electrochemical engineers who design liquid crystal displays in Altoona, but everyone there who buys a cellphone indirectly pays for the time and training of such engineers who live elsewhere.

3. (3 points) Describe the patterns made by these plots. In particular, do cities which are more dependent on these industries have higher productivity?

3. *Are bigger cities more productive, controlling for industry shares?* Using the `gam` function from the `mgcv` package, fit the semi-parametric log-additive model

$$\ln y = \alpha_0 + b \ln N + \sum_{j=1}^{4} f_j(x_j) + \epsilon$$

where y is per-capita GMP, N is population, and x_1 through x_4 are the shares of the four industries.

1. (5 points) Explain how this model is related to, but different than, the power-law scaling model from the last homework. Which terms in the model are parametric, and which are non-parametric?
2. (2 points point) What R command did you use to fit this?
3. (2 points) Report your estimated values for α_0, b, and the residual standard error.
4. (6 points) Provide plots of each of the four partial response functions f_j. Compare them to the plots from question 2 — do they suggest the same relationships between industry shares and the level of productivity, and if not, how do they differ? *Hint*: `help(plot.gam,package="mgcv")`
5. (5 points) Do the residuals seem to have a Gaussian distribution? (Justify your answer.)
6. (5 points) Running `summary` on your fitted model will produce output which includes approximate standard errors and p-values for the parametric terms, assuming homoskedastic Gaussian noise. What standard error and p-value does it report for b? Is that term significant? Do you think you can trust those calculations in this case?

4. *Predictive comparisons*

1. (5 points) Take the fitted power-law scaling model from the last homework. (If you were unable to complete that homework, follow the solutions.) For each city, compute the predicted change in $\ln y$ from increasing that city's population by 10%. Report the average change over all cities.
2. (5 points) Repeat this calculation, for the cities where complete data is available, for the model you fit in Problem 3, assuming that *only* population changes.
3. (5 points) Do the two models seem to lead to different conclusions about the effect of population on productivity? Explain

5. *Model comparisons*

1. (3 points) What is the in-sample mean squared error, for $\ln y$, of the additive model you fit in Problem 3? How much smaller is it than the linear (power law) model from the last homework? Explain why the additive model should always have a smaller in-sample error than the linear model.

   2. (11 points) Describe, concisely and in your own words, a technique for determining whether the additive model from Problem 3 is better able to generalize than the pure power law model. Explain why this technique should be reliable here. (You are free to use a method from 36-401, if you can explain why it is applicable.)
   3. (11 points) Implement this comparison and report your results. Which model is favored?

6. *Evaluation*
   1. (10 points) Based on what you have done so far, does it seem that city size directly effects productivity? Specifically, if an American city wanted to increase its per-capita economic output, should it try to increase population, or change its industries?
   2. (5 points) Suggest additional data, models or comparisons which could improve your analysis.

14

Fair's Affairs

In 1969, the magazine *Psychology Today* did a survey of its readers that included questions about (among other things) how often the respondents had had extramarital sex in the previous twelve months. In 1978 the economist Ray C. Fair used this data to develop a "theory of extramarital affairs" (Fair, 1978)[1], with the idea that people optimize a trade-off between working, spending time with their spouse, and spending time with a "paramour". The model and data have become very well known (there are at least a hundred later papers and books which reference it), and is available as `Affairs` in the package `AER` on CRAN.

The variable `affairs` records the answer to "How often did you engage in extramarital sexual intercourse during the past year", with values of "once a month", or more frequently, all coded as 12. Other variables are sex, age, how many years the respondent had been married[2], whether they had children, how religious they were (on a scale of 1–5), their level of education, how much prestige their occupation had (on a scale of 1–7), and how happy they were with their marriage (on a scale of 1–5).

1. (30 points) *Two specifications*
    1. (15 points) Using logistic regression, fit a model for the *number* of times respondents said they had extramarital sex during the previous year. Describe, in words, the predictions of the model. Which variables are significant predictors?
    2. (15 points) Repeat (1a), but use logistic regression to fit a model for *whether* respondents said they had extramarital sex at all during the previous year.
2. (10 points) Are the same variables significant in both models in problem 1? Do they have the same signs in both models? Should the models match in this way? Explain.
3. (20 points) *Comparing predictions*
    1. (5 points) For each person in the data set, calculate the predicted probability, under both models, that they did *not* have an affair.
    2. (10 points) Plot these against each other. Describe the plot in words.
    3. (5 points) Do the models agree with each other in their predictions? Should they? Explain.

[1] This paper also used a similar survey of readers of *Redbook* in 1974, not part of this data set.
[2] Prof. Fair removed respondents who had never married, or had married more than once.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

4. (20 points) *Calibration*
   1. (2 points) Consider all the people for whom the predicted probability of an affair, according to the model from problem (1a), is less than 10%. What fraction of them report having affairs?
   2. (3 points) Repeat this calculation for predicted probabilities between 10% and 20%, 20% and 30%, etc. Plot the actual frequencies against the predicted probabilities.
   3. (5 points) Make a similar plot for the other model. (You can combine the plots, if the result is clear.)
   4. (10 points) For which model do the predictions seem to match the data best? Explain with reference to your plots.
5. (10 points) Download Fair's paper and read Table I (p. 53). Does it make sense to use a linear response for all of the variables (as in problem 1 above), or would it be better to treat some variables as categorical? Explain.
6. (10 points) *Evaluation*
   1. (5 points) Do either of these models seem to provide an adequate description of the data? (Explain.) If not, what else could one try?
   2. (5 points) Is it reasonable to use this data to develop theories about contemporary behavior? Explain.

15

How the North American Paleofauna Got a Crook in Its Regression Line

Our problem set this week concerns an important question for evolutionary biology and paleontology. It has been argued that larger organisms tend to have selective advantage over smaller ones of the same species, but larger bodies demand more specialized internal structure, more "division of labor", than small ones, indirectly driving the evolution of increased biological complexity (Bonner, 1988). To evaluate this, it is important to know whether species tend to get larger over evolutionary time, and, if so, to characterize this accurately.

Our data set this week is taken from the North American Mammalian Paleofaunal Database, which contains information on the typical body mass of about 2000 living and extinct species of mammals native to North America. (You can find it on the website, `http://www.stat.cmu.edu/~cshalizi/uADA/13/hw/04/nampd.csv`.) Specifically, the columns of the data give: the scientific name of the species; the natural logarithm of its typical body mass (measured in grams); the natural logarithm of the mass of its ancestor (in grams); how long ago it first appeared in the fossil record (in millions of years); and how recently it last appeared (in millions of years; an NA in this column indicates the species is still alive). We will model how the *change* body mass is related to the body mass of the ancestral species. In particular, paleontologists have suggested that the correct model relating change in log mass to ancestral log mass should be piece-wise linear: a downward-sloping line for small ancestral log masses, and flat for larger ancestral masses. In this problem set, you will fit that model, and examine its predictions.

1. (10) *Basics*
    1. (5) Load the data. Create a vector which gives each species' change in log body mass from its ancestor, and add it to the data frame as a new column. Explain, in your own words, what it would mean for a species to have a value of +0.7 in this column. Check that this column has NA values in the correct places. Explain how you know that those are the correct places. Remove all the rows with NA values for the change in log mass, and use this cleaned version of the data for all subsequent parts of the assignment.
    2. (5) Plot the change in log body mass versus ancestral log body mass. Describe the plot briefly.
2. (10) *Linear model*

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

1. (2) Linearly regress the change in log body mass on the ancestral log body mass. Report the coefficients to reasonable precision.
2. (3) Create a new figure which is the scatter-plot from problem 12, plus your fitted regression line.
3. (5) Based on the estimates 21 and the plot from 22, does this model support or undermine the idea that new species tend to be larger than their ancestors? Explain.

3. (15) *Piecewise linear model*
   1. (5) The piece-wise linear model predicts the following mean response as a function of the input x:
   $$\hat{y}(x) = \begin{cases} a + bx & \text{if } x \leq d \\ c & \text{if } x \geq d \end{cases}$$
   Assuming that this is continuous at d, solve for a in terms of b, c and d. Explain why, in this application, it is reasonable to assume continuity.
   2. (10) Write a function in R, called[1] `deac`, that takes in a vector of numbers `x`, and three parameters `b`, `c`, and `d`, and returns the prediction of the model at each value of `x`.
   Check that your `deac` function is working properly by seeing that when $b = -1$, $c = 0.05$ and $d = 2$, giving `x=c(1,1.5,3)` outputs
   ```
   [1] 1.05 0.55 0.05
   ```
   Plot `deac`, with those parameters, as x goes over the range $(0, 4)$. Does it look right?
   Hints: `ifelse` for writing `deac`, `curve` for plotting.

4. (15) Because `deac` varies nonlinearly with parameter d, we cannot estimate it by linear regression. However, we can still estimate the parameters by least squares. To do this, we need to write a function, make a starting guess about the parameters, and use the built-in optimization function `optim` (see recipe 13.2 in *The R Cookbook*).[2]. The following function fits the model to a data set by numerically minimizing the sum of squared errors:

```
my.start <- c(b=-1,c=0.2,d=10)
fit.a.deac <- function(data,start=my.start) {
  sse <- function(par) {
    preds <- deac(data$ln_old_mass,par[1],par[2],par[3])
    sum((data$delta_ln_mass - preds)^2)
  }
  fit <- optim(par=start,fn=sse,method="Nelder-Mead")
  coefficients <- fit$par
  fitted <- deac(data$ln_old_mass,coefficients[1],coefficients[2],
```

[1] From the initials of the scientists who proposed this model; they didn't give it a name.

[2] R has a built-in function, `nls`, for such "nonlinear least-squares" estimation, working more like `lm`. Unfortunately, `nls` can be flaky when the model doesn't have continuous derivatives, which is the case here. Besides, writing your own code builds character.

```
    coefficients[3])
  residuals <- data$delta_ln_mass - fitted
  mse <- mean(residuals^2)
  return(list(coefficients=coefficients,fitted=fitted,residuals=residuals,
    mse=mse,data=data))
}
```

(See online for the commented version; you'll want to source that, rather than typing this in and adding original errors.)

1. (7) Explain what the inner function, `sse`, does.
2. (8) What sort of output does `fit.a.deac` give — a vector, a list, an array, what? What do the various components of the output represent, in terms of the statistical problem?

5. (15) *Starting positions* The code given above looks for a vector of initial parameters called `my.start`, if no other starting point is supplied. The line before the function makes up some values for `my.start`; they are bad ones. We will see in a later problem set that a reasonable guess for d is about 5.
   1. (5) Use this more-reasonable value of d to get a rough guess for c by taking the average change in log mass over all animals whose ancestral log mass exceeds d. Explain why this is a reasonable way to guess at c.
   2. (5) Get a rough guess for b by linearly regressing the change in log mass on ancestral log mass for animals where the ancestral log mass is less than d. Explain why this is a reasonable way to guess at b.
   3. (5) Re-define `my.start` to contain your improved guesses for b, c and d. Run `fit.a.deac` to get a fitted model, which you should call `nampd.deac`. Plot the fitted values as a function of log ancestral mass on a scatter-plot of change in log mass versus log ancestral mass.
6. (20) *Bootstrapping will continue until morale improves.* Use resampling of residuals, not cases, in both parts. *Note:* You can use the same resampled data-frames for both parts of this problem, but it needs more clever programming. 1000 bootstrap replicates takes 1–2 minutes on my computer.
   1. (10) Find bootstrap standard errors, and 95% confidence intervals, for the parameters b, c and d. Report all these quantities.
   2. (10) Find 95% bootstrap confidence bands for the fitted curve, and add them to your plot from problem 53.
7. (15) *Linear vs. Piecewise Linear* One way to compare two models is to see which one can predict the other's parameter values. We will compare the simple linear model from problem 21 with the piecewise linear model `deac` model from problem 53.
   1. (5) Simulate the fitted `deac` model, using resampling of residuals, and estimate the linear model on the simulation. What coefficients do you estimate? Are they compatible with the ones you estimated from the data?

2. (5) Simulate the fitted linear model, using resampling of residuals, and estimate the `deac` model on the simulation. What coefficients do you get? Are they compatible with the ones you estimated from the data?
3. (5) Use five-fold cross-validation to compare the linear model from problem to the piecewise-linear `deac` model. Which one predicts mass changes better?

16

How the *Hyracotherium* Got Its Mass

AGENDA: Using nonparametric smoothing to check parametric models; more practice with simple simulations and function-writing.

We continue to work with the fossil data set from §15. As mentioned there, some paleontologists have suggested that the right curve relating change in log mass to ancestral log mass should be piece-wise linear and homoskedastic: a downward-sloping line for small ancestral log masses, flat for larger ancestral masses, and constant conditional variance:

$$
\begin{aligned}
Y &= \begin{cases} a + bx + \epsilon & \text{if } x \leq d \\ c + \epsilon & \text{if } x \geq d \end{cases} \\
\mathbf{E}[\epsilon|x] &= 0 \\
\mathrm{Var}[\epsilon|x] &= \sigma^2
\end{aligned}
$$

In the last problem set, you fit that model; in this one, you will see whether the data support non-linear corrections.

You will first need to load the data from the other problem set, and add the column of change in log mass to the data frame.

The `mgcv` package is recommended for the additive model in Problem 5. Earlier problems call for spline smoothing, and can be done with either the `smooth.spline` function or with the `gam` function.

1. (10) *Plotting the Parametric Model*
    1. (5) Make a scatter-plot showing the change in log mass as a function of the log ancestral mass.
    2. (5) Add the estimated piecewise linear model from homework 4. You may refer to the solutions for code and parameter estimates, but must explain, in your own words, any code you borrow from there.
2. (25) *Residual inspections*
    1. (5) Calculate the residuals of the estimated piecewise linear model and plot them against the log ancestral mass. Describe any patterns to the plot in words; you should address whether the model systematically over- or under- predicts in certain ranges of ancestral mass, but there may be other important features.
    2. (5) The column `first_appear_Mya` lists how many millions of years ago each

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

species first appeared. Plot the residuals against this variable; describe any patterns.

3. (5) Plot the squared residuals against the log ancestral mass. Add a smoothing spline. Explain whether the scatter-plot and the spline show evidence of heteroskedasticity.
4. (5) Plot the squared residuals against date of first appearance and add a smoothing spline. Explain whether the scatter-plot and the spline show evidence of heteroskedasticity.
5. (5) Plot the histogram of the residuals (not the squared residuals). Are they Gaussian? Should they be, under the model?

3. (10) *A nonparametric alternative*

   1. (7) Fit a spline regression of the change in log mass against log ancestral mass. Plot this spline on the same graph as the data and the estimated piece-wise linear model. Compare, in words, the shape of the spline to that of the parametric model.
   2. (3) Find the in-sample root-mean-square error of both the parametric model and the smoothing spline. Which fits better?

4. (20) *Testing parametric forms*

   1. (3) Write a function to fit the smoothing spline to a data set. Check that it works by making sure it gives the right answer on the original data.
   2. (2) Write a function to calculate the MSE of a fitted smoothing spline. Check that it works by making sure it gives the right answer on the original data.
   3. (5) Write a function to take in a data set and return the difference in MSEs between the parametric model and the smoothing spline. Check that it works by making sure it gives the right answer on the original data.
   4. (5) Write a function to simulate from the estimated piecewise-linear model by resampling the residuals. You can borrow from the solutions to homework 4, but must explain, in your own words, how that code works. How can you check that the simulation works?
   5. (5) Combine your functions to draw 1000 samples from the distribution of this test statistic, under the null hypothesis that the parametric model is right. What is the p-value of this test of the null hypothesis?

5. (25) *Additional Variables* The piecewise linear model implicitly assumes that the relationship between ancestral mass and change in mass is the same at all times. An alternative is that this relationship has itself evolved.

   1. (5) Estimate an additive model which regresses the change in log mass against the log ancestral mass and the date of first appearance. Plot the two partial response functions, and describe, in words, the shape of the curves. Compare the shape of the partial response function for log ancestral mass to the spline curve from Problem 31.
   2. (4) Does the estimated additive model support or undermine the idea that

the relationship between ancestral mass and descendant mass is invariant over time? Explain.

3. (1) What is the in-sample root-mean-square error of the additive model?
4. (10) Explain what you would have to change from your code in Problem 4 to test the piecewise-linear model against the additive model, and what pieces of code could stay the same.
5. (5) Write the new code called for by Problem 54 and run the test. What is the p-value?

6. (10) Is the piecewise-linear, homoskedastic parametric model an acceptable representation of the data? Justify your answer by referring to your work above.

17

How the Recent Mammals Got Their Size Distribution

All of this is shameless ripped off from `http://arxiv.org/abs/0901.0251` but Aaron said it was OK

Problem sets 15 and 16 used regression to study how the typical mass of (mammalian) species changes over evolution: on average new species are heavier than their ancestors, especially if the ancestor was very small, but with a wide variation. If we combine this with the facts that new species branch off from old ones, and that sometimes species go extinct without leaving descendants, we get a model for how the distribution of body masses changes over time. It's not feasible to say much about this model mathematically, but we can simulate it, and check the simulated distribution against the real distribution of body masses today.

The objects in this model are species, each described by its typical mass. (We assume that this does not change over the lifespan of the species.) Each species can produce new species, who mass is related to that of its ancestor according to our previously-learned regression model, or go extinct. As time goes on, the distribution of body masses will fluctuate randomly, but should do so around a steady, characteristic distribution.

More specifically, each species i has a mass X_i, which is required to be between $x_{\min}$, the smallest possible mass for a mammal, and $x_{\max}$, the largest possible mass. At each point in time, one current species A is uniformly selected to evolve into exactly two new species. Each descendant has a mass X_D which depends on the mass of its ancestor, X_A, according to the regression model, plus independent noise:

$$\log X_D = \log X_A + Z + \begin{cases} a + b \log X_A & \text{if } \log X_A \leq d \\ c & \text{if } \log X_A \geq d \end{cases} \tag{17.1}$$

where $Z \sim \mathcal{N}(0, \sigma^2)$. Continuity means that $a = c - bd$; we also need to impose the constraints that $x_{\min} \leq X_D \leq x_{\max}$.

Species become extinct with a probability that depends on their body mass,

$$p_e(x) = \beta x^{\rho} \tag{17.2}$$

Unless otherwise specified, you should use $\sigma^2 = 0.63$; $x_{\min} = 1.8$ grams and $x_{\max} = 10^{15}$ grams; $\rho = 0.025$; $\beta = 1/5000$; and the values of b, c and d from the solutions to Homework 4.

1. (10) Write a function, `rdeac.1`, which takes as inputs a single ancestral mass X_A (not $\log X_A$), the parameters$b$, c, d and σ^2, and the limits $x_{\min}$ and $x_{\max}$. It should generate a candidate value for X_D (not $\log X_D$) from Eq. 17.1 and

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

return it if it is between the limits, otherwise it should discard the candidate value and try again.

1. (2) Set X_A to 40 grams and check, by simulating many times, that the output is always between $x_{\min}$ and $x_{\max}$, even when those values are brought close to 40 grams.
2. (8) Simulate a single X_D value for 100 values of X_A evenly spaced between 1 and 100 grams. Treat this as real data and re-estimate the parameters b, c and d according to the methods of Homework 4; are they reasonably close to those in the simulation?

2. (10) Write a function, `rdeac`, which takes the same inputs as `rdeac.1` *plus* an integer n, and returns a vector containing n independent draws from this distribution. We will test this with $n = 2$, but your code must be more general for full credit.
    1. (4) Check, by simulating, that the first component of the returned vector has the same marginal distribution as the output of `rdeac.1`.
    2. (4) Check that the second component of the returned vector has the same marginal distribution as the first component.
    3. (2) Check that the two components are uncorrelated.
3. (10) Write a function, `speciate`, which takes the same arguments as `rdeac.1`, except that X_A is replaced by a vector of ancestral masses. The function should select one entry from the vector to be X_A, and generate two independent values of X_D from it. One of these should replace the entry for X_A, and the other should be added to the end of the vector.
    1. (2) Check, by simulating, the output always has one more entry than the input vector of masses, no matter how long the input is.
    2. (8) If the input has length n, check that $n - 1$ of the entries in the output match the input.
4. (15) Write a function, `extinct.probs`, which takes as inputs a vector of species masses, an exponent ρ, and a baseline-rate β, and returns the extinction probability for each species, according to Eq. 17.2.
    1. (1) Check that if the input masses are 2 grams and 2500 grams, with the default parameters the output probabilities $\approx 2.0 \times 10^{-4}$ and 2.4×10^{-4} respectively.
    2. (2) Check that if $\rho = 0$, then the output probabilities are always β, no matter what the masses are.
    3. (2) Check that if there input masses are all equal, then the output probabilities are all the same, no matter what ρ and β are.
    4. (10) Write a function, `extinction`, which takes a vector of species masses, ρ and β, and returns a possibly-shorter vector which removes the masses of species which have been selected for extinction. *Hint:* What does `rbinom(n,size=1,prob=p)` do when `p` is a vector of length `n`?
5. (15) *Evolve!*

1. (5) Write a function, `evolve.1`, which takes as inputs a vector of species masses, b, c, d, σ^2, $x_{\min}$, $x_{\max}$, ρ and β, and first does one speciation step, then one round of extinction, and returns the resulting vector of species masses.
2. (5) Write a function, `evolve`, which takes the same inputs at `evolve.1`, plus an integer t, and iterates `evolve.1` t times.
3. (5) How do you know that your functions are working properly?

6. (15) *Re-running history*
   1. (5) Run `evolve` starting from a single species with a mass of 40 grams for $t = 2 \times 10^5$ steps. Save the output vector of species masses as `y1`. Plot the density of `y1`.
   2. (5) Repeat the last step to get a different vector `y2`. Does it have the same distribution as `y1`? How can you tell?
   3. (5) Change the initial mass to 1000 grams and get a vector of final masses `y3`. How does its distribution differ from that of `y1`?
7. (25) The data file MOM_data_full.txt gives the masses of a large (and representative) sample of currently-living species of mammals. The column `mass` gives the mass in grams; the columns `species`, `genus`, `family`, `order` and `code` are identifiers for the particular species, which do not matter to us. Finally, the column `land` is 1 for species which live on land and 0 for those which live in the water.
   1. (5) Load the data and plot the density of masses for land species.
   2. (10) Describe, in words, how the distribution of current species masses compares to that produced by the simulation model in `y1`.
   3. (10) Use the relative distribution method from Chapter F to compare the actual distribution to the distribution of `y1`. Describe the results and what they say about how the data differ from the model.

18

Red Brain, Blue Brain

AGENDA: Practice with density estimation, conditional densities, and classification models.

TIMING: Problems 1–3 and 6 involve fitting models to data, plotting, and interpretation, but no coding. Problem 5 requires doing all that and some bootstrapping, for which you will need to write a little code (along lines you have done before). Problem 7 requires fitting a model and making some plots from it, and you will (probably) need to write a little code, along the lines of examples, to do so. Problem 8 requires comparing models, and you will need to either write some new code, or tweak some example code, to do 82. The solutions to all problems take about 5 minutes to knit without a cache (and about two seconds with a cache — cache everything!).

The data set n90_pol.csv contains information on 90 university students who participated in a psychological experiment designed to look for relationships between the size of different regions of the brain and political views. The variables `amygdala` and `acc` indicate the volume of two particular brain regions known to be involved in emotions and decision-making, the amygdala and the anterior cingulate cortex; more exactly, these are residuals from the predicted volume, after adjusting for height, sex, and similar anatomical variables. The variable `orientation` gives the subjects' locations on a five-point scale from 1 (very conservative) to 5 (very liberal). `orientation` is an ordinal but not a metric variable, so scores of 1 and 2 are not necessarily as far apart as scores of 2 and 3.

Kanai *et al.* (2011)

1. *Marginal density of brain region volumes*
    1. (5) Using `npudens`, estimate the probability density for the volume of the amygdala. Plot it and report the bandwidth.
    2. (5) Repeat this for the volume of the ACC.
2. *Joint density of brain regions*
    1. (5) Using `npudens`, estimate a joint probability density for the volumes of the amygdala and the ACC. What are the bandwidths? Are they the same as the bandwidths you got in problem 1? Should they be?
    2. (5) Plot the joint density. Does it suggest the two volumes are statistically independent? Should they be? You may use three dimensions, color, contours, etc., for your plot, but you will be graded, in part, on how easy to read it is.
3. *Predicting brain sizes from political views*
    1. (10) Using `npcdens`, find the conditional density of the volume of the amygdala as a function of political orientation. (Make sure that you are treating

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

`orientation` as an ordinal variable.) Report the bandwidths. Is the bandwidth for the amygdala the same as either of the previous two bandwidths you have found for it? Should it be? Plot the distribution, and comment on whether it suggests any relationship between the size of this brain region and political orientation.

2. (5) Repeat this for the conditional density of the ACC as a function of orientation.

4. *Creating a binary response variable*
   1. (1) Create a vector, `conservative`, which is 1 when the subject has `orientation` ≤ 2, and 0 otherwise.
   2. (2) Explain why the cut-off was put at an `orientation` score of 2 (as opposed to some other cut-off).
   3. (1) Check that your `conservative` vector has the proper values, *without* manually examining all 90 entries.
   4. (1) Add `conservative` to your data frame. (Creating a new data frame with a new name will only get you partial credit.)
5. *Logistic regression*
   1. (5) Fit a logistic regression of `conservative` (not `orientation`) on `amygdala` and `acc`. Report the coefficients to no more than three significant digits. Explain what the coefficients mean.
   2. (5) Using case resampling, give bootstrap standard errors and 95% confidence intervals for the coefficients. Was the restriction to three significant digits reasonable?
6. (10) *Generalized additive model* Fit a generalized additive model for `conservative` on `amygdala` and `acc`. (Be sure to smooth both the input variables.) Make sure you are using a logistic link function. Report the intercep. Plot the partial response functions, and explain what they mean (be careful!).
7. *Kernel conditional probability estimation*
   1. (5) Using `npcdens`, find the conditional probability of `conservative` given `amygdala` and `acc`. Make sure `npcdens` treats `conservative` as a categorical variable and not a continuous one. Report the bandwidths.
   2. (5) Plot the estimated conditional probability that `conservative` is 1, with `acc` set to its median value and `amygdala` running over the range $[-0.07, 0.09]$. (The plotting range for `amygdala` exceeds the range of values found in the data.) *Hint:* your code will need to provide values for `acc`, for `amygdala` *and* for `conservative` (why?).
   3. (5) Plot the estimated conditional probability that `conservative` is 1, with `amygdala` set to its median value and `acc` running over the range $[-0.04, 0.06]$. (This plotting range also requires extrapolating outside the data.)
8. *Classification* The models from problems 5–7 predict probabilities for `conservative`. If we have to make a definite prediction of whether someone is conservative or not, we should predict 1 if the probability is ≥ 0.5 and 0 otherwise.

1. (7) Find such predictions for each subject, under each of the three models. What fraction of subjects are mis-classified? What fraction would be mis-classified by "predicting" that none of them are conservative?
2. (8) Re-calculate the classification error rates using leave-one-out cross-validation for each model.

19

Brought to You by the Letters D, A and G

AGENDA: Identifying and estimating causal effects; the importance of selecting appropriate controls; estimating effects in non-linear models.

TIMING: Problems 1 and 2 are straightforward data manipulation; problem 3 needs you to fit a linearly model and bootstrap some standard errors; problems 4 and 5 need you to fit nonparametric models, extract predictions from them, and bootstrap some standard errors; problem 6 needs you to take the ratio of two covariances, and bootstrap some standard errors. Despite all the bootstrapping and using kernel regressions, the solutions take less than two minutes to knit (without a cache). Problems 2–6 all require you to think about some graphical models. Problem 7 requires you to do some math.

The file `sesame.csv` contains data on an experiment which sought to learn whether regularly watching *Sesame Street* caused an increase in cognitive skills, at least on average. The experiment consisted of randomly selecting some children, the treated, and *encouraging* them to watch the show, while others received no such encouragement. The children were tested before and after the experimental period on a range of cognitive skills. (Table 19.1 lists the variables.)

1. *Before and after* (5) For each of the skills variables, find the difference between pre-test and post-test scores, and add the corresponding column to the data frame. Name these columns `deltabody`, `deltalet`, etc. Describe and run a check that the values in these columns are at least approximately right (without examining them all).
2. *Naive comparison*
    1. (2) Find the mean `deltalet` scores for children who were regular watchers, and for children who were not regular watchers. Provide standard errors in these means as well, and the standard error for the difference in means.
    2. (3) What must be assumed for the difference between these means to be a sound estimate of the average causal effect of switching from not watching to regularly watching *Sesame Street*? Is that plausible? Suggest a way the assumption could be tested.
3. *"Holding all else constant"*
    1. (5) Linearly regress the change in reading scores on regular watching, and all other variables except `id`, `viewcat`, and the `post`-tests.Report the coefficients and bootstrap standard errors to *reasonable* precision. (Be careful of categorical variables.)
    2. (3) Explain why `id`, `viewcat`, and the `post` variables had to be left out of the regression. (The reasons need not all be the same.)

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

3. (2) What would someone who had only taken linear regression report as the average effect of making a child become a regular watcher of *Sesame Street*?
4. (5) What would we have to assume for this to be a valid estimate of the average causal effect? Is that plausible?

4. Consider the graphical model in Figure 19.1.
    1. (10) Find a set of variables which satisfies the back-door criterion for estimating the effect of regular watching on `deltalet`.
    2. (5) Do a nonparametric regression of `deltalet` on `regular` and the variables you selected in 41. (You can use any nonparametric method you like; you may need to be careful about which variables are categorical.) Find the corresponding estimate of the average effect of causing a child to become a regular watcher. Give a bootstrap standard error for this average treatment effect.

5. Consider the graphical model in Figure 19.2.
    1. (5) There is at least one set of variables which meets the back-door criterion in Figure 19.2 which did not meet it in Figure 19.1. Find such a set, and explain why it meets the criterion in the new graph, but did not meet it in the old one.
    2. (5) Explain whether or not the set of control variables you found in 41 still works in the new graph.
    3. (5) Do a nonparametric regression of `deltalet` on `regular` and the variables you selected in 51. Find the corresponding estimate of the average effect of causing a child to become a regular watcher, and a bootstrap standard error for this average treatment effect.
    4. (5) Find a pair of variables which are conditionally (or marginally!) independent in Figure 19.1 but are not in Figure 19.2, and vice versa. Explain why. *Note:* Both the conditioned and conditioning variables must be observed; the point is to find something which could be checked with the data.

6. *Instrumental encouragement* Some children were randomly selected for encouragement to watch *Sesame Street*. This is encoded in the variable `encour`.
    1. (3) Explain why `encour` is a valid instrument for the effect of regular watching on `deltalet` in Figure 19.1. Do you need to control for anything else?
    2. (2) Explain why `encour` is a valid instrument in Figure 19.2. Do you need to control for anything?
    3. (5) Describe a DAG in which `encour` would *not* be a valid instrument, even though it is randomized by the experimenters.
    4. (5) Estimate the average effect on `deltalet` of causing a child to become a regular watcher using `encour` and the Wald estimator (see notes). Provide a standard error using bootstrapping.

7. (5) Do Exercise 20.2.

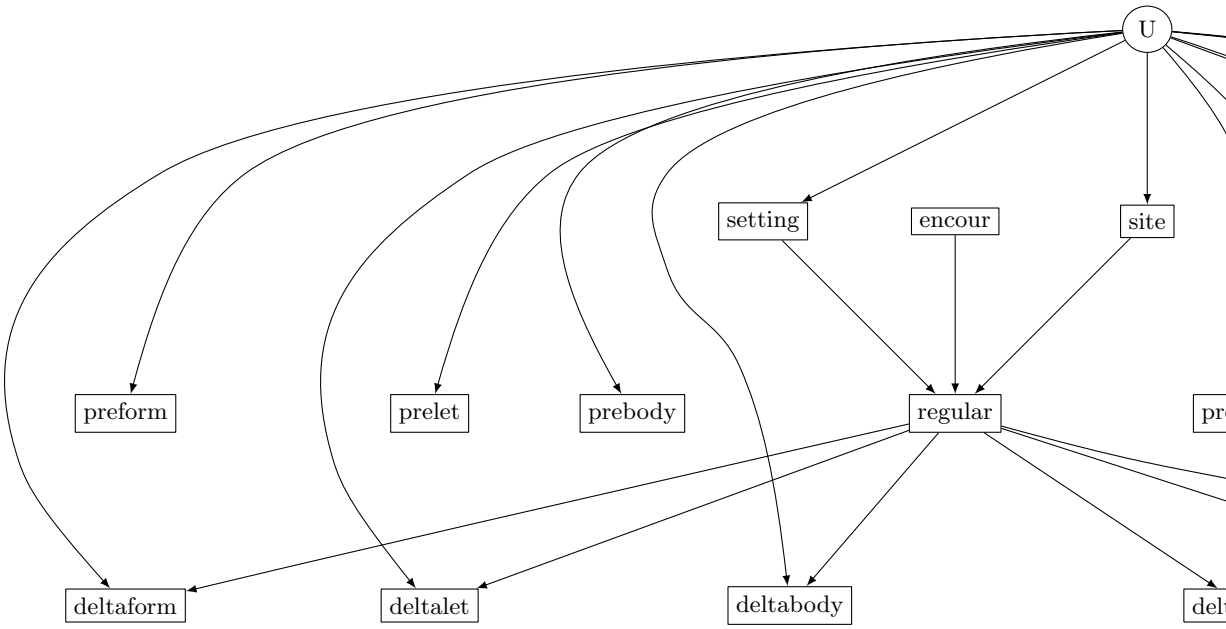


Figure 19.1 First DAG.

```
## [1] 80
```

Extra credit (5) Test whether either of the two conditional independence relations from 54 hold in the data.

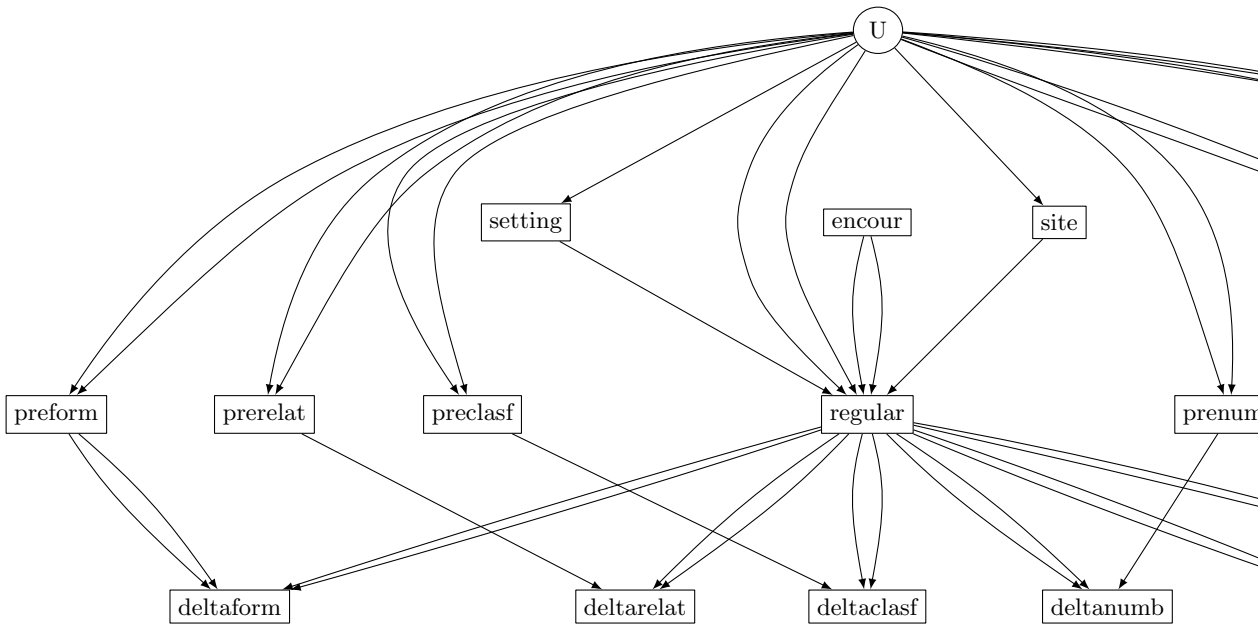


Figure 19.2 Second DAG.

`id`	subject ID number
`site`	categorical; social background
	1: Disadvantaged inner-city children, 3–5 yr old
	2: Advantaged suburban children, 4 yr old
	3: Advantaged rural children, various ages
	4: Disadvantaged rural children
	5: Disadvantaged Spanish-speaking children
`sex`	male=1, female=2
`age`	in months
`setting`	categorical; whether show was watched at home (1) or school (2)
`viewcat`	categorical; frequency of viewing *Sesame Street*
	1: watched < 1/wk
	2: watched $1 - -2$/wk
	3: watched $3 - -5$/wk
	4: watched > 5/wk
`regular`	0: watched < 1/wk, 1: watched ≥ 1/wk
`encour`	encouraged to watch = 1, not encouraged=0
`peabody`	mental age, according to the Peabody Picture Vocabulary test
	(to measure vocabulary knowledge)
`prelet`, `postlet`	pre-experiment and post-experiment scores on knowledge of letters
`prebody`, `postbody`	pre-test and post-test on body parts
`preform`, `postform`	pre-test and post-test on geometric forms
`prenumb`, `postnumb`	tests on numbers
`prerelat`, `postrelat`	tests on relational terms
`preclasf`, `postclasf`	pre-test and post-test on classification skills
	(“one of these things is not like the others”)
	(“one of these things just doesn’t belong”)

Table 19.1 *Variables in the* `sesame` *data file. The pre- and post- experiment test scores are integers, but can be treated as continuous.*

20

Teacher, Leave Those Kids Alone! (They're the Control Group)

AGENDA: Applying causal-inference ideas in an experimental setting. Practice in thinking through what variables should and should not be controlled for.

TIMING: None of the problems here should require very elaborate coding or time-consuming computations.

The Tennessee STAR project was a randomized experiment which sought to determine whether children learn more in classrooms with fewer students. Students within participating schools were randomly assigned to small (¡ 18 student) classrooms, to ordinary-sized classrooms, and to ordinary classrooms where the teacher had an aide. The study began in kindergarten and continued through third grade. Students initially assigned to the small-class condition for the most part stayed in it (there were a few unavoidable exceptions for administrative reasons); students assigned to the two large-class conditions were re-randomized in the second year of the study, and thereafter changed only minimally. New students entering the schools in the study were randomized into the three conditions. Teachers were also randomized as to which kind of classroom they got. Learning was assessed (in the initial phase of the project) through annual standardized tests of reading and math.

A standard version of the data set is available as `STAR` in the `AER` package, which you may need to install. See `help(STAR)` for the definitions of the variables named below.

General: Whenever you are asked to give standard errors, you should either bootstrap or provide an explanation of why, in this particular situation, R's default calculations of standard errors should be reliable. Unless explicitly called for, do not report R's p-values, or any significance stars.

1. *Causality? Reverse causality?*
    1. (5) Linearly reqgress `readk` and `mathk` on `stark`. Report the coeffcients and standard errors. Explain why a non-parametric regression would be redundant here.
    2. (5) Linearly regress `read3` and `math3` on `stark`. Report the coefficients and their standard errors as above.
    3. (5) Explain how a randomized treatment received in kindergarten can predict test scores three years later.
    4. (5) Linearly regress `readk` and `mathk` on `star3`. Report the coefficients and their standard errors as above.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

5. (5) Explain how a treatment received in the third grade can predict test scores in kindergarten, three years *earlier*.
6. (5) To estimate the causal effect of the `stark` on `readk` and `mathk`, should we control for `star3`? (Explain.)
7. (5) To estimate the causal effect of the `star3` on `read3` and `math3`, should we control for `stark`? (Again, explain.)

2. (15) For each year from kindergarten through third grade, provide an estimate of the expected reading and math scores when students are assigned to a regular classroom, a small classroom, and a regular classroom with a teacher's aide. Include an estimated standard error for each of these. You may present your results either as a table or graphically; make sure it's easy to read and compare across conditions.

   Explain how you obtained your estimates, and why that procedure is, for this data, a valid way of estimating the desired causal effect. If you have to control or adjust for any covariates to get the causal effects, explain which ones you used and why.
3. (15) *Heterogeneity of effects* There is considerable interest in knowing whether the effects of smaller classes are different for different groups of students.

   1. (10) Report estimates of the effect of the three classroom sizes on kindergarten reading and math scores, for all six ethnic sub-groups in the data. Include standard errors.
   2. (5) Explain why, to get such estimates from linear regression, the right models would be of the form `lm(readk~stark*ethnicity)`, and why `lm(readk~stark+ethnicity)` would be uninformative.

4. (25) *Observational inference in an experimental study* Students whose families are sufficiently poor qualify for free lunches at school. This is recorded in the variables `lunchk` through `lunch3`. We want to know whether being above or below this threshold level of poverty has a causal effect on student's scores.

   1. (5) Report the mean scores for reading and for math in each grade for students who do and do not qualify for free lunches (in that grade). Include standard errors.
   2. (5) If we want to estimate the effect of `lunchk` on kindergarten reading and math scores, does it make sense to control for `stark`? Explain.
   3. (10) Consider the following variables: `gender`, `ethnicity`, `schoolk`, `experiencek`, `tethnicityk`, `systemk`, `schoolidk`, `lunch1`. When estimating the effect of `lunchk` on kindergarten test scores, which of these should be controlled for, which of them should not be controlled for, and which of them do you not have enough information to say? If you answer "not enough information" for any variables, what more would you have to know? (Be more specific than "the complete causal graph".)
   4. (5) If we want to estimate the effect of `lunchk` on first-grade reading and math scores, under what assumptions should we control for `readk` and `mathk`? Under what assumptions should we not control for them?

```
## [1] 130
```

21

Estimating with DAGs

This homework will illustrate some of the advantages of using a known DAG structure. You will need to read the lectures on graphical models carefully in order to do it.

Figure 21.1 is an elaboration of the graph used in lectures. All problems refer to it, unless otherwise specified.

The file `fake-smoke.csv` contains some (synthetic) data, for use in problem 5.

1. *Parents and children* (10 points)
    1. (5 points) For each variable in the model, list its parents; or, if it has no parents, say so.
    2. (5 points) For each variable in the model, list its children. (Some variables have no children.)
2. *Joint distributions and factorization* (10 points) Using the graph, list the smallest collection of marginal and conditional distributions which must be estimated in order to get the joint distribution of all variables.
3. *Associations* (20 points) Should there be a positive association, a negative association, or no association between the following variables? Explain with reference to the graph. (2 points each)
    1. Yellowing of teeth and cancer?
    2. Yellowing of teeth and cancer, controlling for smoking?
    3. Yellowing of teeth and cancer, controlling for occupational prestige?
    4. Yellowing of teeth and cancer, controlling for smoking and exposure to asbestos?
    5. Smoking and cancer, controlling for the amount of tar in the lungs?
    6. Asbestos and cancer, controlling for cellular damage?
    7. Smoking and cancer, controlling for asbestos?
    8. Smoking and asbestos, controlling for cellular damage?
    9. Tar in lungs and cancer, controlling for asbestos, smoking, and yellowing of teeth?
    10. Smoking and cancer, controlling for asbestos and occupational prestige?
4. *Using conditional independence to specify regressions* (40 points)
    1. (10 points) We wish to know the conditional risk of cancer given smoking. What other variables should be controlled for? Which other variables do not need to be controlled for?

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

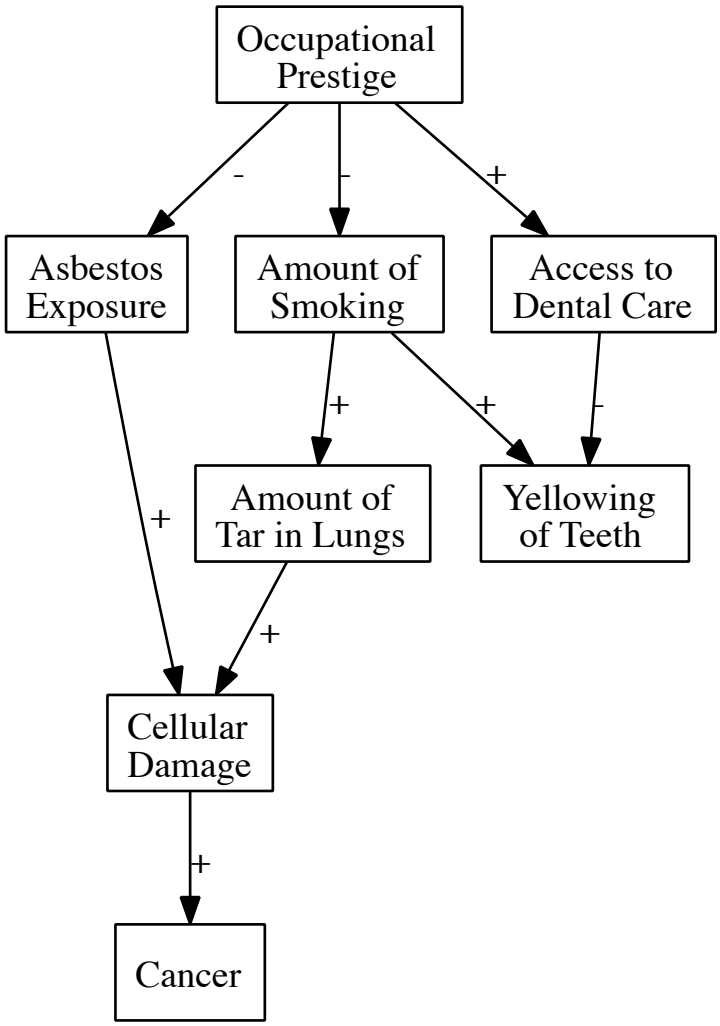


Figure 21.1 Graphical model for use in all problems, except part of the last. Signs on arrows indicate the sign of the associations (not necessarily linear) between parents and children.

2. (10 points) Using the `fake-smoke.csv` data from the class website, fit a logistic regression model for the risk of cancer given the level of smoking, controlling for any appropriate covariates.
3. (10 points) Using the same data set, fit another logistic regression for the

risk of cancer using *all* the covariates. What does this say about the relationship between smoking and cancer? Why is this different than what is implied by the model in 4b?

4. (5 points) A medical insurance company needs to predict the risk of cancer among customers in order to set rates. Should it use the model from 4b or the one from 4c? Why? (Assume, for the sake of the problem, that the training data and the insurance customers are both representative samples of the general population.)
5. (5 points) A doctor wants to advise their patients about what actions to take to reduce their risk of cancer. Should they use the model from 4b or 4c? Why?

5. (20 points) Consider the alternative graph in Figure 21.2.
    1. (10 points) Repeat problem 3 with the new graph. Clearly indicate in your response which associations differ for the two DAGs.
    2. (10 points) Suggest an experiment, or an observational analysis, which could let us check which structure was right; explain, in terms of the graphs.
6. (10 points) Extra Credit: Which DAG did the example data come from? How can you tell?

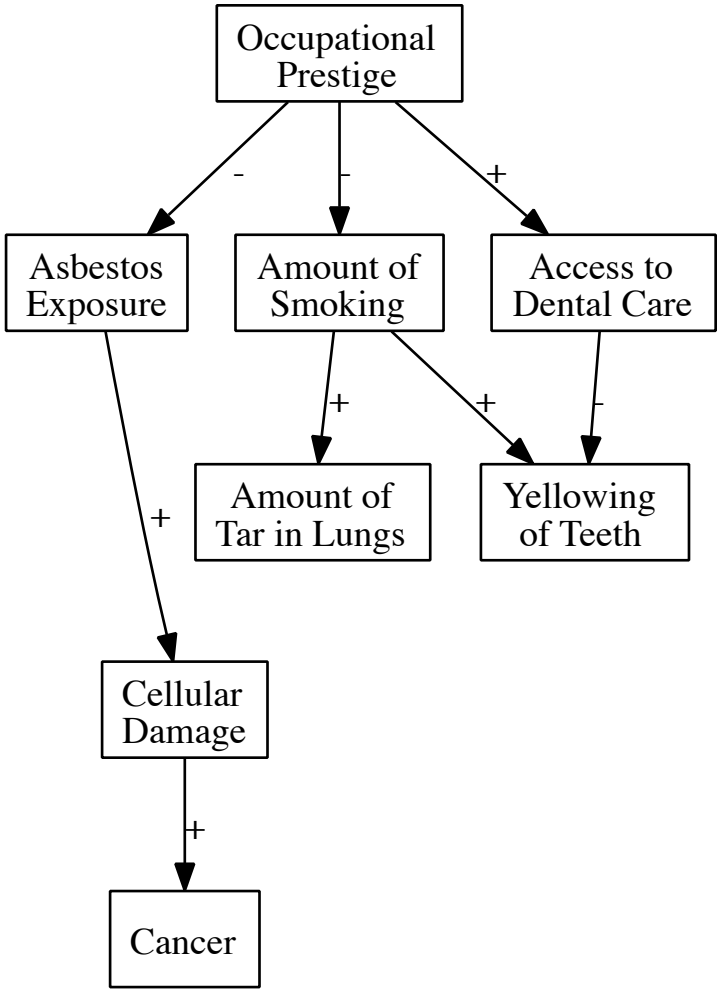


Figure 21.2 An alternative DAG for the same variables.

22

Use and Abuse of Conditioning

1. (30 points) Refer to figure [[1]] in Problem Set 21.
   1. (5 points) Using the back door criterion, describe a way to estimate the causal effect of smoking on cancer.
   2. (5 points) Using the front door criterion, describe a *different* way to estimate the causal effect of smoking on cancer.
   3. (5 points) Is there a way to use instrumental variables to estimate the causal effect of smoking on cancer in this model? Explain.
   4. (5 points) Using your back-door identification strategy and the data file from last time, estimate $\Pr(cancer = 1 | do(smoking = 1.5))$.
   5. (5 points) Repeat this using your front-door identification strategy.
   6. (5 points) Do your two estimates of the casual effect match? Explain.
2. (25 points) Take the model in Figure 22.1. Suppose that $X \sim \mathcal{N}(0,1)$, $Y = \alpha X + \epsilon$ and $Z = \beta_1 X + \beta_2 Y + \eta$, where ϵ and η are mean-zero Gaussian noise with common variance σ^2. Set this up in R and regress Y twice, once on X alone and once on X and Z. Can you find any values of the parameters where the coefficient of X in the second regression is even approximately equal to α? (It's possible to solve this problem exactly through linear algebra instead.)
3. (25 points) Take the model in Figure 22.2 and parameterize it as follows: $U \sim \mathcal{N}(0,1)$, $X = \alpha_1 U + \epsilon$, $Z = \beta X + \eta$, $Y = \gamma Z + \alpha_2 U + \xi$, where ϵ, η, ξ are independent Gaussian noises with mean zero and common variance σ^2. If you regress Y on Z, what coefficient do you get, on average? If you regress Y on Z and X? If you do a back-door adjustment for X? (Approach this either analytically or through simulation, as you like.)
4. (20 points) Continuing in the set-up of the previous problem, what coefficient do you get for X when you regress Y on Z and X? Now compare this to the front-door adjustment for the effect of X on Y.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

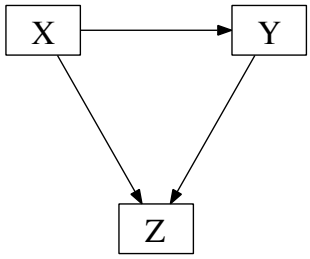


Figure 22.1 DAG for problem 2.

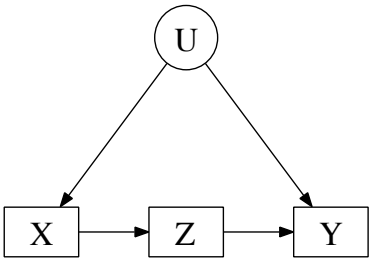


Figure 22.2 DAG for problems 3 and 4.

23

What Makes the Union Strong?

Finding the factors which control the frequency and severity of strikes by organized workers is an important problem in economics, sociology and political science[1]. Our data set, `http://www.stat.cmu.edu/~cshalizi/uADA/12/hw/06/strikes.csv`, kindly provided by a distinguished specialist in the field, contains information about the incidence of strikes, and several variables which are plausibly related to that, for 18 developed (OECD) countries during 1951–1985:

Source: Western (1996)

- Country name
- Year
- Strike volume, defined as "days [of work] lost due to industrial disputes per 1000 wage salary earners"
- Unemployment rate (percentage)
- Inflation rate (consumer prices, percentage)
- "parliamentary representation of social democratic and labor parties". (For the United States, this is the fraction of Congressional seats held by the Democratic Party.)
- A measure of the centralization of the leadership in that country's union movement, on a scale of 0 to 1[2].
- Union density, the fraction of salary earners belonging to a union (only available from 1960).

[[TODO: Fix point assignments]]

Note that some variables are missing (`NA`) for some cases.

1. Estimate a linear model to predict strike volume in terms of all of the other variables, *except* country and year.
    1. Report the coefficients, with 90% (not 95%) confidence intervals calculated according to
        1. (2) The standard formulas
        2. (9) Resampling of the residuals
        3. (9) Resampling of the cases

        Do not use more digits than you can justify.

[1] Or it used to be, anyway.

[2] This measure really should be a constant for each country over the period, but having a variable with only 8 levels is trouble for the spline smoother used in Problem 3, so a very small amount of artificial noise (± 0.005 at most) has been added to each value.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

   2. (10) Describe the meaning of the coefficients *qualitatively*. (I.e., do not write "A one unit change in foo produces a change of bar units in strike volume" over and over.)
   3. (5) Rank the predictor variables from most to least important, with "importance" measured by the magnitude of the predicted change to strike volume in response to a 1% relative change of the predictor away from its mean value.
   4. (5) Rank the predictor variables from most to least important in terms of predicted response to a 1 standard deviation change in the variable.
   5. (5) Do the two rankings agree? Should they? Which one seems more reasonable for this problem?

2. Some theories suggest that English-speaking countries have legal and political institutions which make strikes operate differently than in other industrialized countries. Figure out which countries in the data set are primarily English-speaking, create an indicator (dummy) variable for whether a case belongs to one of those countries, and add it to the data set.

   1. (5) Fit a linear model in which the predictors from Problem 1 interact with the English-using variable. Report the new coefficients (to *reasonable* precision)
   2. (5) Explain how (if at all) this model differs qualitatively from the model in Problem 1.
   3. (5) Use five-fold cross-validation to compare this model to the model in Problem 1. Which one does better?

3. Fit an additive model for strike volume as a smooth function of all the variables except country and year.

   1. (5) Plot all the partial response functions. Do they agree qualitatively with the conclusions you drew from the model in Problem 1?
   2. (5) Consider increasing each of the predictor variables by 1% from its mean, leaving the other variables alone. Rank the predictors according to the magnitude of this model's predicted change in strike volume. Would the ranking be the same for a 1% decrease? *Hint:* use `predict` and a data frame with artificial data.
   3. (5) Consider increasing each of the predictor variables by one standard deviation from its mean, leaving the other variables alone. Rank the predictors according to the magnitude of this model's predicted change in strike volume.
   4. (5) Discuss the contrast (if any) between these rankings, and the corresponding ones for the linear model.

4. (10) Use the methods of Chapter 10 to test whether the linear model from Problem 1 is well-specified against an additive alternative.
5. *Continuing past the training data*

   1. (2) What were the values of unemployment, inflation, union density, and

`left.parliament` for the United States in 2009? *Hint*: You can get most of these from the last *The Statistical Abstract of the United States*.

2. (4) Assuming the union centralization variable for the US in 2009 was 0, what strike volume was predicted by (i) the model from problem 1, (ii) the English-is-different model from problem 2, and (iii) the additive model from problem 3?
3. (4) The actual strike volume for the United States in 2009 was 0.8. Is this plausible under any of the models? *Hint:* How much do you expect actual values to differ from predicted values?

6. 1. (5) Use `pc()` from `pcalg` to obtain a graph, assuming all relations between variables are linear. Report the causal parents (if any) and children (if any) of every variable. If the algorithm is unable to orient one or more of the edges, report this, and in later parts of this problem, consider all the graphs which result from different possible orientations.
   Note: See `http://bactra.org/weblog/914.html` for help with installing `pcalg`. The most troublesome component is the `Rgraphviz` package. If you are unable to get `Rgraphviz` to work, you can still extract the information from the fitted model returned by `pc`: if that's `pc.fit`, then `pc.fit@graph@edgeL` is the "edge list" of the graph, listing, for each node, the nodes it has arrows *to*. With this information, you can make your own picture of the DAG.
   2. (10) Linearly model each variable as a function of its parents. Report the coefficients (to reasonable precision), the standard deviation of the regression noise (ditto), and 95% confidence intervals for all of these, as determined by bootstrapping the residuals.
   3. (10 total) You should find that strike volume and union density are not connected, but that there is at least one directed path linking them — either density is an ancestor of strike volume, or the other way around.
      1. (5) Find the expected change in the descendant from a one-standard-deviation increase in the ancestor above its mean value.
      2. (5) Linearly regress the descendant on all the other variables, including the ancestor. According to this regression, what is the expected change in the descendant, when the ancestor increases one SD above its mean value and all other variables are at their mean values?
   4. (15 total) Check the linearity assumption for each variable which has a parent. (Putting in interactions and/or quadratic terms is inadequate and will result in only partial credit at best.)
      1. (5) Describe your method, and why it should work.
      2. (5) Report the p-value for each case, to reasonable precision.
      3. (5) What is your over-all judgment about whether it is reasonable to model each endogenous variable as linearly related to its parents? If you need more information than just p-values to reach a decision, describe it.

5. (10) Discuss the over-all adequacy of the model, on both statistical grounds (goodness-of-fit, appropriateness of modeling assumptions, etc.) and substantive, scientific ones (whether it makes sense, given what is known about the processes involved).

24

An Insufficiently Random Walk Down Wall Street

[[TODO: Brad DeLong (!) points out by e-mail that one should really define returns here as $\log((P_{t+1} + D_{t+1})/P_t)$, where D is the dividend series — prices aren't the only thing that matters with the S&P! Obtain a historical dividend series, or a dividend-adjusted price series.]]

In this assignment, you will work with a data set of historical values for the S&P 500 stock index. You will need to download `SPhistory.short.csv` from the class website. This data set records the actual *prices* of the index, say P_t on day t, but in finance we actually care about the returns, $\frac{P_t}{P_{t-1}}$, or about the logarithmic returns,

$$R_t = \log \frac{P_t}{P_{t-1}}$$

since we care more about whether we're making 1% on our investment than \$1 per share. In this assignment, "returns" always means "logarithmic returns".

Problems 2 and 3 are about estimating the first percentile of the return distribution, $Q(0.01)$, under various assumptions. The returns will be larger than this 99% of the time, so $Q(0.01)$ gives an idea of how bad the bad performance will be, which is useful for planning. Note that a calendar year contains about 250 trading days, and so should average two or three days when returns are even worse than $Q(0.01)$. Problems 4 and 5 are about predicting future returns from historical returns, and the uncertainty in this. Doing all the bootstrapping for problem 5 may be time-consuming, and should not be left to the last minute.

1. (5) Load the data file, take the last column (containing the daily closing price), and calculate the logarithmic returns. Note that the file is in reverse chronological order (newest first). When you are done, if everything worked right, running `summary` on the returns series should give

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.094700 -0.006440  0.000467 -0.000064  0.006310  0.110000
```

   Hint: `help(rev)` and Recipe 14.8 in *The R Cookbook*.
2. In finance, it is common to model daily returns as independent Gaussian variables.
   1. (5) Find the mean and standard deviation of the returns. What is $Q(0.01)$ of the corresponding Gaussian distribution? *Hint:* `qnorm`.
   2. (5) Write an expression which will generate a series of independent Gaussian values of the same length as the returns, with the mean and standard deviation you found in 21. Check that the mean and standard deviation of the output is *approximately* right, and that their histogram looks like a bell-curve.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

3. (10) Write a function which takes in a data vector, calculates its mean and standard deviation, and returns $Q(0.01)$ according to the corresponding Gaussian distribution. Check that it works by seeing that it matches what the answer you got in 21 when run on the actual returns.
4. (10) Using the code you wrote in 22 and 23, find a 95% confidence interval for $Q(0.01)$ from 21. *Hint*: Look at the examples of parametric bootstrapping in Chapter 6.
5. (5 points) What is the first percentile of the data? Is it within the confidence interval you found in 24? *Hint:* `quantile`.

3. 1. (5) Use `hist` to plot the histogram of returns. Also plot, on the same graph, the probability density function of the Gaussian distribution you fit in problem 21. Comment on their differences.
   2. (5) Write a function to resample the returns; it should generate a different random vector of the sample length as the data every time it is run. Check that running `summary` on these vectors produces results close to those on the data. *Hint:* Look at the examples of resampling in Chapter 6.
   3. (5) Write a function to calculate $Q(0.01)$ from an arbitrary vector, *without* assuming a Gaussian distribution. Check that it works by seeing that its answer, when run on the real data, matches what you found in 25.
   4. (10) Using the code you wrote in 32 and 33, find a 95% confidence interval for $Q(0.01)$. Compare this to your answer in 24. Which is more believable, and why? *Hint:* Look at the examples in the notes of non-parametric bootstrapping.
4. (10) Using `npreg`, fit a kernel regression of R_{t+1}, tomorrow's returns, on R_t, today's returns. (Use the automatic bandwidth selector.) Report the selected bandwidth and the in-sample mean-squared error. Make a scatter-plot with R_t on the horizontal axis and R_{t+1} on the vertical axis, and add the estimated kernel regression function. Comment on the shape of the curve. *Hints:* Make a data frame with R_t as one column and R_{t+1} as another column. Also, see examples in Chapter 4 of plotting fitted models from `npreg`.
5. (25) *Uncertainty in the kernel regression*
   1. (5) Write a function which resamples (R_t, R_{t+1}) pairs from the returns series, and produces a new data frame of the same size as the original. Check that it works by running `summary` on it, and seeing that both columns *approximately* match the summaries of the data. *Hint:* look at the examples of resampling cases for regression in the notes.
   2. (10) Write a function which takes a data frame with appropriately-named columns, and runs a kernel regression of R_{t+1} on R_t. It should return fitted values at 30 evenly-spaced values of R_t which span its observed range.
   3. (10) Using your code from 51 and 52, add 95% confidence bands for the kernel regression to your plot from problem 4. *Hint:* See the examples of plotting bootstrapped nonparametric regressions in the notes.

[[TODO: Integrate this version with the one above]]

1. (5 points) Load the data file, take the last column (containing the daily closing price), and calculate the logarithmic returns. Note that the file is in reverse chronological order (newest first). When you are done, if everything worked right, running `summary` on the returns series should give

```
     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
-0.094700 -0.006440  0.000467 -0.000064  0.006310  0.110000
```

2. In many applications in finance, it is common to model daily returns as independent Gaussian variables.
    1. (5 points) Use maximum likelihood to estimate the mean and standard deviation of the best-fitting Gaussian, and the $Q(0.01)$ it implies.
    2. (5 points) Write a function which simulates a data set of the same size as the real data, using the independent Gaussian model you fit in (2a), and returns a list or vector, with components named `mean` and `sd`, containing the parameter values estimated from the simulation output.
    3. (5 points) Write a function which takes as arguments a list or vector, with components named `mean` and `sd`, and returns the first percentile of the corresponding Gaussian distribution. Check that it works by verifying that when run with mean 5 and sd 2, it returns 0.347.
    4. (10 points) Using the code you wrote in (2b) and (2c), find a 95% confidence interval for $Q(0.01)$ from (2a). *Hint*: Look at the examples in the notes of parametric bootstrapping.
    5. (5 points) What is the first percentile of the data? Is it within the confidence interval you found in (2d)?

[[TODO: Clarify using quantile here.]]

3. 1. (5 points) Use `density()`, or any other suitable non-parametric density estimator, to plot the distribution of returns. Also plot, on the same graph, the Gaussian distribution you fit in problem 2. Comment on their differences.
    2. (10 points) Write a function to re-sample the returns, and calculate $Q(0.01)$ on each surrogate data set. Use this to find a 95% confidence interval for $Q(0.01)$. *Hint:* Look at the examples in the notes of non-parametric bootstrapping.
4. (15 points) In an **autoregressive** model, the measurement at time t is regressed on the measurement at time $t-1$, $X_t = \phi_0 + \phi_1 X_{t-1} + \epsilon_t$. (§23.4 has much more information.) Use `lm` to fit an autoregressive model to the returns. Give the estimates of ϕ_0, ϕ_1 and $\mathbb{V}[\epsilon]$, and try to interpret what they mean. Also give the reported standard error for $\widehat{\phi_1}$.
5. *Hint*: Look at the examples in the notes of re-sampling regression residuals.
    1. (5 points) Write a function which re-samples the residuals of the autoregressive model from (4). Make sure it returns a vector of values. Check that the mean and standard deviation of its output are close to those of the residuals.

2. (15 points) Write a function which simulates the autoregressive model you fit in (4), with noise provided by the function you wrote for (5a). The initial value of X should match the initial value in the data, and it should return a vector.
3. (5 points) Write a function which takes a time series, fits an autoregressive model, and returns the estimate of ϕ_1. Check that it works by seeing that when it's give the data, the output matches what you found in (4).
4. (10 points) Using the function you wrote in (5c), and the simulator you wrote in (5b), find the bootstrap standard error for $\widehat{\phi_1}$. Does it match what `lm` reported in (4)?

Note: If you cannot solve (5b), you can get full credit for (5d) using the built-in function `arima.sim` instead, but make sure that the distribution of innovations or noise comes from the function you wrote in (5a). If you cannot solve (5a), you can get full credit for (5b) and (5d) by providing suitable Gaussian noise.

25

Predicting Nine of the Last Five Recessions

The data set `http://www.stat.cmu.edu/~cshalizi/uADA/13/exams/3/macro.csv` on the class website contains five standard macroeconomic time series for the United States, from the beginning of 1948 to the beginning of 2010: total national income or GDP; value of goods consumed; investment spending; hours worked; and output per hour worked for all non-financial firms. (Some of these series are in inflation-adjusted dollars, some of them are in hours, and some of them are indexes where a particular date has been set as 100 and others are expressed relative to that.) All variables are measured "quarterly", i.e., four times a year.

Most macroeconomic forecasting models do not concern themselves directly with these values, but only with the logged fluctuations around their long-run trends.

For full credit on the modeling questions, you must use models which go beyond those available in 401, *or* you must use appropriate methods to show that linear model are justified here.

It is first necessary to remove trends; macroeconomists traditionally do this with the following function.

```
hpfilter <- function(y, w=1600){
 eye = diag(length(y))
 d = diff(eye,d=2)
 ybar = solve(eye + w*crossprod(d), y)
 yhat = log(y) - log(ybar)
 return(list(fluctuation=yhat,trend=ybar))
}
```

1. (10) Create five plots, showing each of the variables and its trend (as returned by `hpfilter`) as functions of time. Use a logged scale for the vertical axis. Report R^2, with and without logging, for each of the five trends.
2. (10) Plot the logged fluctuations around trend (as returned by `hpfilter`) for each of the five variables. Does it make sense to compare these fluctuations across variables? Do the fluctuations look stationary? — After this problem, references to the variables always mean their logged fluctuations around their trends.
3. (10) Are the variables Gaussian? (You can do better than looking at a histogram.)
4. (20) For the first four variables (GDP, consumption, investment, hours worked),

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

fit an additive regression of each variable on the values of all four at the previous time-step. Use only data up to, but not including, 2005 ("the training period"). Report the mean squared error on the training data (to reasonable precision), and include plots of the partial response functions. Describe, in words, what the partial response functions say about the relations between these variables.

5. (20 total) Using the circular block bootstrap, with blocks of length 24, generate new time series which are as long as the training data.
    1. (4) Write a function to calculate the mean squared errors of the fitted models from Problem 4, on a time series. (Each of the four variables should have its own MSE.) Check that it works by making sure that it gives the right answer for the training data.
    2. (6) Report the mean MSEs, and the standard error of these means, from enough bootstrap replicates that the standard errors are no more than 10% of the means.
    3. (10) What do you need to assume for the numbers from 52 to be good estimates of the generalization error of this model?
6. (20 total) "Real" (as opposed to "monetary") business cycle theories hold that fluctuations in macroeconomic variables are ultimately caused by exogenous "real shocks", especially changes to productivity. The productivity variable in `macro.csv` is a measurement of this variable, which, according to these theories, should be exogenous. The other variables, in such theories, are endogenous.
    1. (10) Fit an model for each of the four endogenous variables, as an additive function of the endogenous variables in the previous quarter, and productivity for the previous four quarters. Report the MSEs and include plots of the partial response functions. Compare the plots to those in Problem 4.
    2. (4) Describe a method which could be used to decide whether including productivity in this way really improves predictive performance. Discuss the assumptions of the method, and why you think they apply here.
    3. (6) Implement your method. For which variables does including productivity actually help? How confident are you of this conclusion?
7. (10 total) Now consider the period 2005–2010. What are the mean squared errors, on this data, of
    1. (4) Predicting according to the additive model from Problem 4?
    2. (4) Predicting according to the additive model from Problem 6?
    3. (2) Predicting the mean of each variable, as estimated from the training period?
8. (5, extra credit) Explain how what `hpfilter` does is related to spline smoothing.

26

Debt Needs Time for What It Kills to Grow In

An important and controversial question in macroeconomics and political economy is whether high levels of government debt causes the economy to grow more slowly or even shrink. There are several plausible-sounding reasons why it might[1]; some economists claim that there is a threshold level of debt, perhaps around 90% of GDP, above which growth rates plummet.

Against this, there are other reasons why high levels of debt might *not* cause growth to slow, at least not always[2]. In particular, since "high levels of government debt" are defined relative to the size of the economy, as a high ratio of debt to GDP, slow growth itself might cause higher levels of government debt.

This week's data set contains information on GDP and government debt for a selection of countries since World War II. For each country and year, we should have the GDP (nominal, i.e., not adjusted for inflation or differences in exchange rates) and the size of government debt (also nominal). Unfortunately, one or both values may be missing for some countries in some years.

1. (10) The data set contains a variable, `growth`, which is the annual growth rate in real (inflation-adjusted) GDP for each country and year. It also contains a variable, `ratio` which is the ratio of government debt to GDP. Make a scatterplot with `growth` on the vertical axis and `ratio` on the horizontal. Describe the patterns you see, if any.
2. (15) Run a nonparametric regression of `growth` on `ratio`, and plot the resulting curve. Describe and interpret the curve. Does it suggest an abrupt slowing of growth above some threshold level of debt?
3. (10) Since changes in government debt levels might take some time to affect economic growth, we would like to compare growth in year $t+1$ to `ratio` in year t. Create a new variable, `growth.lead1`, which records for each country/year

[1] High levels of government borrowing might "crowd out" investing in the private sector, by using up available savings and/or raising the interest rates at which businesses can borrow; capitalists might anticipate that the debt will either be paid off through high taxes or discharged through inflation, and prefer to spend their money on luxuries now, rather than invest and see the investment go away later; high levels of debt might lead to lower confidence that the government generally knows what it's doing, making investment seem too risky; etc.

[2] A depressed economy has unused resources, so government employment needn't lead to crowding out; the things government spends money on (roads, schools, hospitals, basic research, honest markets) increase the value of private investments; governments which can borrow large sums are receiving a market endorsement of their willingness and ability to pay their debts; etc.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/

the *next* year's GDP growth, with NAs in the right places when it is not available. Describe, in words, how your code works. Add `growth.lead1` to the data frame.

Hints: Make sure that you do not confuse growth rates from different countries (so that, e.g., the last year for Austria gets a growth rate from Belgium). You may find Recipes 14.7 (and 6.6) from *The R Cookbook* helpful.

4. (10) Plot `growth.lead1` against `ratio`, and do a nonparametric regression of the former on the latter. Describe the results, and compare them to those of Problem 2.
5. (15) Economic growth rates tend to be rather persistent over time within countries. Estimate an additive model where `growth.lead1` is predicted from `growth` and `ratio`. Is the partial response to the previous year's growth nearly linear? Should it be? Compare the partial response function for debt to the curves from problems 2 and 4.
6. (10) Create a new variable, `growth.lag1`, which represents the *previous* year's growth rate (with NAs in appropriate places), and add it to the data set. Plot it against `ratio` and fit a nonparametric regression. Does `ratio` do a better job of predicting `growth` or `growth.lag1`?
7. (15) Estimate an additive model in which the current year's `ratio` is predicted by last year's `ratio`, last year's `growth`, and the current year's `growth`. (You may have to create a new column.) Describe the partial response functions, and whether any predictor variables could be dropped.
8. (15) Explain what we would have to assume for the model in Problem 5 to give us an unconfounded estimate of the causal effect of government debt on future economic growth; be as specific as possible. (You may want to draw some DAGs, and include them in your write-up.) Comment on how plausible those assumptions are, and on what might go wrong if the assumptions fail.

27

How Tetracycline Came to Peoria

Now-common ideas like "early adopters" and "viral marketing" grew from sociological studies of the diffusion of innovations. One of the most famous of these studies tracked how a then-new antibiotic, tetracycline, spread among doctors in four towns in Illionis in the 1950s (Coleman *et al.*, 1957). In this exam, we will go back to that data to look at one of the crucial ideas, that of the innovation (prescribing tetracycline) spreading from person to person.

For this assignment, you will need two data files, ckm_nodes.csv and ckm_network.dat.[1] The former has information about each individual doctor in the four towns. `adoption_date` records the month in which the doctor began prescribing tetracycline, counting from November 1953. If the doctor did not begin prescribing it by month 17, i.e., February 1955, when the study ended, this is recorded as `Inf`. If it's not known when or if a doctor adopted tetracycline, their value is `NA`. (Apparently no doctors gave up tetracycline after adopting it.) Other columns record when the doctor attended medical school, whether they attend medical conferences (and if so, what kind), how many medical journals they read, and other information about the individual doctors. Note that the covariates in this file are a mix of ordinal variables, categorical variables, and numerical variables.

[[TODO: Better URLs]]

The `ckm_network.dat` file contains a binary matrix, which records the social network among the doctors. There is one row and one column for each doctor; the i, j entry is 1 if doctor number i and doctor number j knew each other, and 0 if they did not.

[[TODO: Re-work points and instructions for this not to be an exam?]]

1. (5) Create a plot of the number of doctors who *began* prescribing tetracycline each month versus time. (It is OK for the numbers on the horizontal axis to just be integers rather than formatted dates.) Produce another plot of the *total* number of doctors prescribing tetracycline in each month. (The curve for total adoptions should first rise rapidly and then level out around month 6.)
2. Estimate the probability that a doctor who had not yet adopted the drug will begin to do so in a given month t, as a function of the *total* number of doctors N_t who had adopted *before* t. (You may assume that these probabilities are the same for all t.) You may estimate this function however you like, but be sure to explain how you are estimating these probabilities, and how you know that method is reliable *in this particular case*. (This may involve model checking.)

[1] Slightly modified from `http://moreno.ss.uci.edu/data.html` to fit R conventions, and collapsing three distinct, directed social relationships into one undirected social network.

11:43 Friday 23rd February, 2024
Copyright ©Cosma Rohilla Shalizi; do not distribute without permission
updates at `http://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/`

   1. (5) Report these probabilities as a curve, with N ranging from 0 to 125. If you do not think you can estimate the whole range, plot as much as you can, and explain why you cannot go further. For full credit, your plot must have more than 17 points. Also for full credit, your curve should be accompanied by some measure of its error.
   2. (5) Averaging over doctors and months, how much does the predicted probability of adoption change N increases by 1? Give a standard error to this change in predicted probabilities.

   Hint: You may find it useful to create a new data frame which records, for each month, the number of doctors who adopted tetracycline that month, and the number who had previously adopted tetracycline.
3. Estimate the probability that a doctor i who had not yet adopted the drug will begin to do so in month t, as a function of the number C_{it} of *doctors linked to* i who had adopted before t. (Again, you may assume that these probabilities are the same for all t.)
   1. (8) Make a plot of these probabilities, with C_{it} ranging from 0 to 30. If you do not think you can estimate the whole range, plot as much as you can, and explain why you cannot go further. For full credit, your plot must include at least 29 points, and include a measure of uncertainty in your estimates. Does your curve support the idea that the use of tetracycline is transmitted from one doctor to another through the social network? Explain, including a description of what curves which did *not* support this idea would look like, or why the shape of this curve is actually irrelevant to this issue.
   2. (7) Averaging over doctors and months, how much does the predicted probability of adoption change when C_{it} increases by one? What is your standard error for this change in predicted probabilities?

   Hint: You may find it useful to create a data frame recording, for every combination of doctor and month, whether that doctor began prescribing tetracycline that month, the number of their contacts who began prescribing before that month. Such a data frame should have 2125 rows.
4. 1. (1) Are your estimates from problem 22 and 32 consistent with one another? Explain.
   2. (4) What would you have to assume for either of these to be estimates of the causal effect on adoption by other doctors of making one extra doctor adopt the drug? Be as specific as you can, rather than just repeating definitions from the notes. Drawing graphs is encouraged.
5. Estimate a model which predicts the probability that a doctor i who had not yet adopted the drug by month t will begin to do so in month t, as a function of C_{it} and of the covariates which indicate when i went to medical school, whether they attended medical-society meetings (and if so what kind), and how many medical journals they read.
   1. (5) Plot the estimated probability of adoption as a function of C_{it} for doctors who read the minimal number of journals, do not attend conferences, and graduated from medical school (i) in 1919 or earlier, (ii) in the 1920s,

and (iii) in 1940 or after. For full credit, have all three lines on the same plot (clearly visually distinct from each other), and some measure of uncertainty for each line.

2. (5) Averaging over doctors and months, how much does increasing C_{it} by one change the probability of doctor i adopting tetracycline in month t? Include a standard error for this change in predicted probabilities.
3. (5) Under what assumptions does this give a valid estimate of the average causal effect of increasing C_{it} by one?

Note If you want to display the social network, the R package `igraph` is designed for such things.

27.1 Formatting Instructions and Rubric

Your main report should be a humanly-readable document of at most 10 single-spaced pages, including figures. It should have the following sections:

INTRODUCTION describing the scientific problem and the data set, possibly including *relevant* summary statistics or exploratory graphs. (Do not include EDA just to have EDA.)

SPECIFIC PROBLEMS answering the questions set above, but avoiding the check-list, itemized format in favor of continuous text, with a logical succession of sentences and paragraphs. (Writing coherently is more important than following the order of the questions.)

CONCLUSIONS summarizing what you have learned from the data and models about whether the transmission of an innovation from person to person is really a good description of how these doctors came to use tetracycline.

You may assume that the reader has a general familiarity with the contents of 401, and with the models and methods we have covered so far in the course, but will need to be reminded of any details. The reader should not be assumed to have any prior familiarity with the data set.

Code

All statistical results must be supported by appropriate code, or they will receive no credit. ("Show your work.") Code should only appear in the text of the report when it is the best way of conveying some point. The ideal would be to use R Markdown, or knitr+LaTeX, to embed all computations in a humanly readable document, and submit both the knitted version and the source[2] As a second best, it is acceptable to submit a PDF document containing all text and figures, and a separate .R file, containing all supporting computations, clearly labeled via the comments so that it is easy to see which claims or results go with which pieces of code.

[2] See examples at `http://yihui.name/knitr/demos/`, and the useful chunk options like `echo` at `http://yihui.name/knitr/options/`; also the examples in the solutions to exam 1.

Rubric

As usual, this describes the ideal.

Words

(5) The text is laid out cleanly, with clear divisions and transitions between sections and sub-sections. The writing itself is well-organized, free of grammatical and other mechanical errors, divided into complete sentences logically grouped into paragraphs and sections, and easy to follow from the presumed level of knowledge.

Numbers

(5) All numerical results or summaries are reported to suitable precision, and with appropriate measures of uncertainty attached when applicable.

Pictures

(5) Figures and tables are easy to read, with informative captions, axis labels and legends, and are placed near the relevant pieces of text.

Code

(15) The code is formatted and organized so that it is easy for others to read and understand. It is indented, commented, and uses meaningful names. It only includes computations which are actually needed to answer the analytical questions, and avoids redundancy. Code borrowed from the notes, from books, or from resources found online is explicitly acknowledged and sourced in the comments. Functions or procedures not directly taken from the notes have accompanying tests which check whether the code does what it is supposed to. All code runs, and the Markdown file knits (if applicable). The main text of the report is free of intrusive blocks of code, which are used only when a specifically-computational point is being made, or when code is actually the clearest way of describing a point.

Inference and Uncertainty

(10) The actual estimation of model parameters or estimated functions is technically correct. All calculations based on estimates are clearly explained, and also technically correct. All estimates or derived quantities are accompanied with appropriate measures of uncertainty (such as confidence intervals or standard errors).

Conclusions

(10) The substantive questions about diffusion of innovations are all answered as precisely as the data and the model allow. The chain of reasoning from estimation results about models, or derived quantities, to substantive conclusions is both clear and convincing. Contingent answers ("if X, then Y, but if Z, then W") are likewise described as warranted by the model and data. If uncertainties in the

data and model mean the answers to some questions must be imprecise, this too is reflected in the conclusions.

Extra credit

(10) Up to ten points may be awarded for reports which are unusually well-written, where the code is unusually elegant, where the analytical methods are unusually insightful, or where the analysis goes beyond the required set of analytical questions. *Example:* Simulating the model estimated in problem 5, taking the set of doctors who have adopted in month 1 for the initial conditions and continuing for another 16 months, with a detailed and quantitative comparison of multiple simulation runs to the actual data, and an informative assessment of what the comparison says about the strengths and weaknesses of the model.